Eurographics Workshop on 3D Object Retrieval (2015)
I. Pratikakis, M. Spagnuolo, T. Theoharis, L. Van Gool, and R. Veltkamp (Editors)

3D GrabCut: Interactive Foreground Extraction for Reconstructed 3D Scenes

Gregory P. Meyer and Minh N. Do

University of Illinois at Urbana-Champaign

Abstract

In the near future, mobile devices will be able to measure the 3D geometry of an environment using integrated depth sensing technology. This technology will enable anyone to reconstruct a 3D model of their surroundings. Similar to natural 2D images, a 3D model of a natural scene will occasionally contain a desired foreground object and an unwanted background region. Inspired by GrabCut for still images, we propose a system to perform interactive foreground/background segmentation on a reconstructed 3D scene using an intuitive user interface. Our system is designed to enable anyone, regardless of skill, to extract a 3D object from a 3D scene with a minimal amount of effort. The only input required by the user is a rectangular box around the desired object. We performed several experiments to demonstrate that our system produces high-quality segmentation on a wide variety of 3D scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Over the past few years, low-cost depth cameras, such as Microsoft's Kinect, have become increasingly more common. By fusing multiple depth images, it is possible to reconstruct a high-quality 3D model of a scene [NDI*11]. Therefore, 3D models can be generated without the need of a skilled artist or high-cost 3D scanner. In the near future, depth sensing technology will be integrated into mobile devices, enabling anyone to construct a 3D model of their environment [Goo14, Occ14]. Unlike 3D models built by an artist or 3D scanner in a controlled environment, a 3D reconstruction of a natural scene may contain a desired foreground object and an unwanted background region. For that reason, we propose a system to perform interactive foreground extraction on a reconstructed 3D scene using an intuitive user interface (Figure 1).

Several mesh segmentation algorithms have been proposed based on hierarchical clustering [GWH01], k-means [STK02], spectral clustering [LZ04], core extraction [KLT05], primitive fitting [AFS06], region growing [JLCW06], random walks [LHMR08], and graph cuts [KT03, GF08, FL*11]. The majority of mesh segmentation techniques are designed to decompose a 3D model into visually similar parts. These parts could be surface patches with similar curvature, or they could correspond to semantic parts of the object. Segmenting a mesh into multiple parts can be useful for several applications including texture mapping, mesh simplification, mesh editing, shape matching, morphing, and animation [Sha08]. For our problem, we want to segment a 3D model into two regions, foreground and background, and each region may or may not contain a collection of visually similar parts. As a result, we cannot rely on these previous proposed methods.

Ji et al. [JLCW06] and Fan et al. [FL*11] proposed techniques for segmenting a 3D model into foreground/background regions. Both of these methods use a stroke-based user interface, which can require trial-and-error to obtain a high-quality segmentation.

In the field of image processing, Rother et al. [RKB04] developed the GrabCut technique which substantially reduces the amount of user interaction to segment an image. Instead of requiring the user to draw strokes to identify the foreground and background regions, the user simply needs to draw a bounding box around the foreground object. Inspired by work of Rother et al. [RKB04], we propose 3D GrabCut, an interactive method for segmenting a 3D model into foreground/background regions based on a rough 2D bounding box drawn around the foreground object. To our knowledge, this is the first time this type of user interface has been used to segment a 3D model.

DOI: 10.2312/3dor.20151048

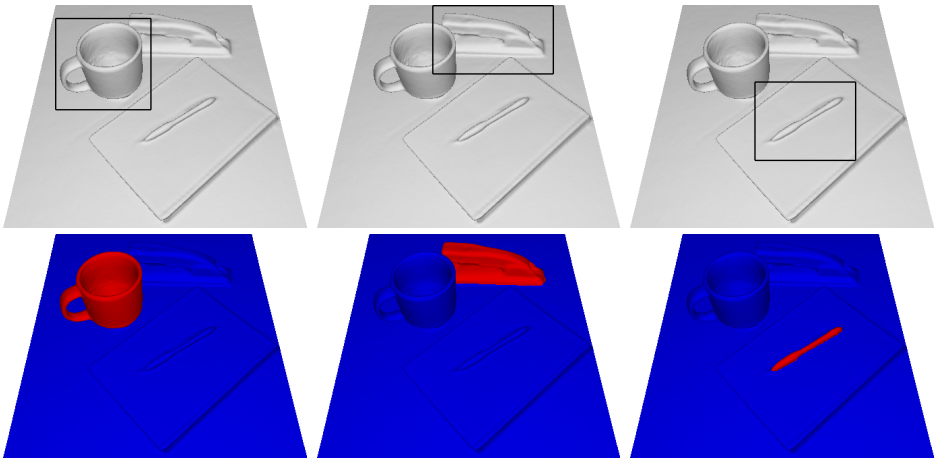

Figure 1: Our proposed method can efficiently extract an object from a reconstructed 3D scene with only a small amount of user interaction (a rectangular box around the desired object). The top row shows the reconstructed 3D scene with user input. The results of the segmentation are shown in the bottom row.

In the following sections we describe our proposed method (Section 2) and demonstrate qualitatively the accuracy and the usability of our approach (Section 3).

2. Our approach

Given a reconstructed 3D scene, our goal is to decompose the scene into foreground and background regions based on a rough segmentation provided by the user.

The 3D scene is represented by a triangular mesh $M = \{V, E, F\}$, which is a collection of vertices $V = \{v_i | 1 \leq i \leq n\}$, edges $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V, i \neq j\}$, and faces $F = \{f_{ijk} = (v_i, v_j, v_k) | v_i, v_j, v_k \in V, i \neq j, i \neq k, j \neq k\}$ that describe the scene's geometry [Sha08]. For our application, we want to partition F into foreground $\mathcal{F}$ and background $\mathcal{B}$ regions, where $\mathcal{F}$ contains the entire foreground object and $\mathcal{B}$ contains everything else ($\mathcal{B} = \neg\mathcal{F} \cap F$).

In this section, we describe the interface that enables a user to approximately identify the foreground object, as well as, our method for segmenting the mesh based on the user's input.

2.1. User interface

After loading a triangular mesh, our system enables the user to freely navigate around the 3D scene. To specify the desired object, the user simply draws a rectangular box around it. The 2D bounding box is projected from the image plane into the scene creating a 3D bounding volume $\mathcal{V}$. Mesh faces inside the volume are added to the foreground region $\mathcal{F} = \{f | f \in \mathcal{V}, f \in F\}$, and faces outside the volume are added to the background region $\mathcal{B} = \{f | f \notin \mathcal{V}, f \in F\}$. Our mesh segmentation algorithm refines this initial segmentation until only the desired object remains in the foreground region. Figure 2 illustrates a 3D volume generated from a 2D bounding box, as well as, the corresponding initial segmentation (the bounding box used is shown in the left column of Figure 1).

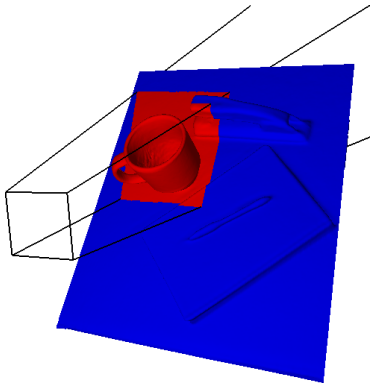

Figure 2: An example of a 3D volume created by projecting a 2D bounding box from the image plane into the 3D scene. The triangular faces inside the volume are initially labeled as foreground, and everything else is labeled as background.

A user can provide multiple bounding boxes from different viewpoints, and each bounding box will have a corresponding 3D volume within the 3D scene. Only faces that lie within the intersection of all the 3D volumes are marked as foreground. However, most scenes only require a single bounding box to accurately segment the mesh.

With stroke-based user interfaces, a user must consider which parts of the mesh to label foreground and which parts to label background [JLCW06]. Often, it is not immediately obvious which parts of the mesh should be labeled, and it

may require trial-and-error to obtain a high-quality segmentation. On the other hand, our user interface is highly intuitive. Without much deliberation, a user can acquire a precise segmentation of the mesh.

2.2. Mesh segmentation

The user provides a 2D bounding box around the object he/she wants to extract. The bounding box defines a 3D volume, which we use to initialize the foreground $\mathcal{F}$ and background $\mathcal{B}$ regions. Our goal is to shrink the foreground region so that it only contains the desired object.

Instead of working with the mesh directly, we build a face-adjacency dual graph $G = \{V_D, E_D\}$ [Sha08]. Two faces $f \in F$ and $f' \in F$ are considered adjacent $(f, f') \in \mathcal{A}$, if they share a common edge $e \in E$. For each triangular face in the mesh, there is a corresponding vertex in the dual graph $V_D = \{f | f \in F\}$; in addition, adjacent faces are connected by an edge in the dual graph $E_D = \{(f, f') | (f, f') \in \mathcal{A}\}$.

In order to extract the desired object, the mesh segmentation is posed as an optimization problem. We utilize an energy function $\mathbf{E}$ whose minimum corresponds to a good segmentation of the mesh [BJ01],

$$\mathbf{E}(L) = \sum_{f \in V_D} U(l_f) + \gamma \sum_{(f,f') \in E_D} V(l_f, l_{f'}) \qquad (1)$$

where $L = \{l_f | f \in V_D\}$ is a label map indicating whether a face f is assigned as foreground ($l_f = 1$) or background ($l_f = 0$). The data term U defines the penalty for assigning a specific label to a face, the smoothness term V specifies the penalty for assigning a different label to adjacent faces, and γ designates the relative importance of the smoothness term versus the data term. The assignment of labels that minimizes the energy function,

$$L^* = \arg\min_L \mathbf{E}(L) \qquad (2)$$

can be determined efficiently with the graph cuts algorithm [BJ01]. We experimentally found that setting $\gamma = 150$ produces good results for all of our reconstructed 3D scenes.

2.2.1. Smoothness term

The smoothness term V penalizes adjacent faces for being assigned different labels. To obtain a good segmentation of the mesh, the penalty $V(f, f')$ should be significant when f and f' lie on the same object. Therefore, we defined the smoothness term as follows:

$$V(l_f, l_{f'}) = \left| l_f - l_{f'} \right| \cdot \exp\left(-\frac{D(f, f')}{\sigma_V} \right) \qquad (3)$$

where $D(f, f')$ measures the distance between the two faces. The distance function,

$$D(f, f') = \alpha \cdot \frac{d_\Delta(f, f')}{\langle d_\Delta \rangle} + (1 - \alpha) \cdot \frac{d_\theta(f, f')}{\langle d_\theta \rangle} \qquad (4)$$

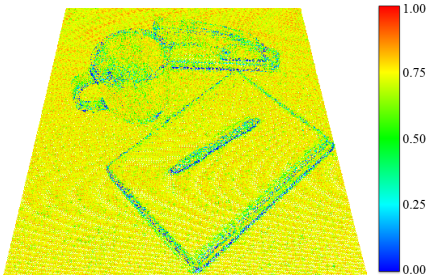

Figure 3: Visualization of the smoothness term.

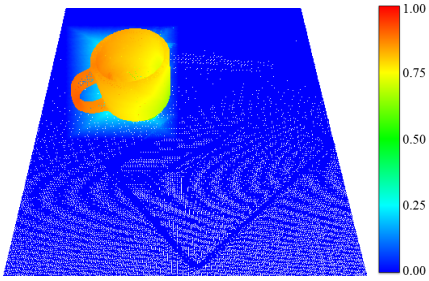

Figure 4: Visualization of the data term.

is a combination of the geodesic distance d_Δ and angular distance d_θ between f and f'. $\langle d_\Delta \rangle$ and $\langle d_\theta \rangle$ denote the average geodesic distance and angular distance, respectively. The geodesic distance is computed by adding the distance between each face's center of mass and the midpoint of their shared edge [STK02]. The angular distance is based on the dihedral angle between the two faces,

$$d_\theta(f, f') = \mu \left[1 - \cos\left(\text{dihedral}(f, f') \right) \right]. \qquad (5)$$

According to the minima rule [HR84], the human visual system defines object boundaries along negative minima of curvature. Therefore, to penalize segmentation boundaries that do not lie on a concave crease, we set $\mu = 0.1$ for convex edges and $\mu = 1.0$ for concave edges [LHMR08]. Also, we place a greater weight on the angular distance by setting $\alpha = 0.25$. Figure 3 visualizes the smoothness term for the scene shown in Figure 1. (To illustrate the smoothness term, we set $\left| l_f - l_{f'} \right| = 1$ for all pairs of adjanceny faces.)

2.2.2. Data term

The purpose of the data term U is to define the penalty for assigning a face one label over another. For 2D images, GrabCut uses Gaussian mixture models to model the color distribution in the foreground and background regions, and their data term is defined based on these models [RKB04]. For example, if a pixel has a high probability of being foreground based on the mixture models, then labeling the pixel as background would incur a large penalty. For mesh segmentation, texture information may or may not be available, so we cannot use a data term that relies on the color distribu-

tion to differentiate the foreground/background regions. Additionally, we cannot leverage surface curvature to discriminate between the regions because both regions may contain shapes with similar curvature. The only reliable information we have about the regions is the bounding box provided by the user.

Based on the user's input, we know that every face outside the volume $\mathcal{V}$, should be labeled background; therefore, labeling a face in $\mathcal{B}$ as foreground should incur a high penalty. The faces inside $\mathcal{V}$ may belong to the foreground object or the background region. As mentioned above, humans define object boundaries along negative minima of curvature [HR84]. If we assume the only object completely within $\mathcal{V}$ is the desired foreground object, then there should be a path between the boundary of the foreground object and the boundary of $\mathcal{V}$ that does not contain any concave creases. Therefore, the path distance between a face in the background region and the boundary of $\mathcal{V}$ should be small, and the path distance between a face on the foreground object to the boundary of $\mathcal{V}$ should be large, where the distance between adjacent faces is defined by Equation 4. The greater the path distance between a face in $\mathcal{F}$ and the boundary of $\mathcal{V}$, the larger the penalty should be for labeling it as background. Accordingly, we defined the data term as follows:

$$U(l_f) = \begin{cases} (1-l_f)\cdot\left[1-\exp\left(-\frac{G(f)}{\sigma_U}\right)\right] & \text{if } f \in \mathcal{F} \\ l_f \cdot K & \text{if } f \in \mathcal{B} \end{cases} \quad (6)$$

where K is a large constant, and $G(f)$ measures the geodesic distance between a face and the boundary of $\mathcal{V}$ along the shortest path normalized by the average path distances from all faces in $\mathcal{F}$ to the boundary of $\mathcal{V}$,

$$G(f) = \frac{g(f)}{\langle g \rangle}. \quad (7)$$

The function $g(f)$ determines the distance along the shortest path $P = (f_0, f_1, \ldots, f_n)$ from f to the closest face in $\mathcal{B}$,

$$g(f) = \min_P \sum_{i=0}^{n-1} D(f_i, f_{i+1}) \quad (8)$$

where $(f_i, f_{i+1}) \in E_D$, $f_0 = f$, $f_n \in \mathcal{B}$, and $f_i \in \mathcal{F}\big|_{i=0}^{n-1}$. We use Dijkstra's shortest path algorithm to efficiently compute $g(f)$. Figure 4 visualizes the data term corresponding to the 3D scene and bounding volume shown in Figure 2. (To illustrate the data term, we set $l_f = 0$ for all faces.)

2.2.3. Iterative refinement

Occasionally, part of the background remains in the foreground region after segmentation. Similar to 2D GrabCut [RKB04], we iteratively refine the segmentation by repeatedly solving Equation 2. After each iteration, we update the foreground and background regions based on the label assignment that minimizes the energy function $\mathbf{E}$ ($\mathcal{F} = \{f|l_f^* = 1\}$ and $\mathcal{B} = \{f|l_f^* = 0\}$). Furthermore, we recompute the data term based on the updated foreground and background regions. The number of iterations depends on how tightly the bounding box is defined. For most 3D scenes, only one or two iterations are required for our method to converge to a good segmentation.

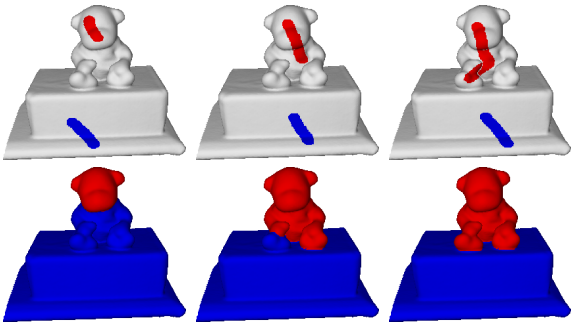

Figure 5: Stroke-based methods can require trial-and-error to obtain a good segmentation.

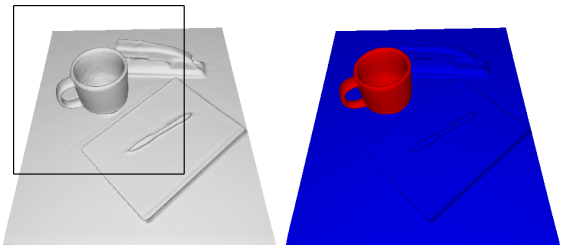

Figure 6: Our technique is insensitive to the position of the bounding box, as long as, the desired object is the only object completely contained within the bounding box.

3. Results

We evaluated our technique on several 3D scenes reconstructed by a consumer depth camera (Microsoft's Kinect), and the results are shown in Figure 7. The first three columns in Figure 7, represent a typical scene with the object the user wants to model placed on top of a flat surface. Although this arrangement is trivial to segment with our proposed method, stroked-based methods can still require trial-and-error as seen in Figure 5. For stroke-based techniques, it is often unclear how much of the mesh needs to be labeled. Our approach has the benefit of being highly intuitive, the user simply places a box around the object he/she would like to extract. The last two columns in Figure 7, demonstrate the flexibility of our algorithm. Even with a subtle object boundary or noisy background, our method can correctly extract the desired object from the 3D scene. In addition, our method is not sensitive to placement of the bounding box provided by the user as shown in Figure 6. As long as the

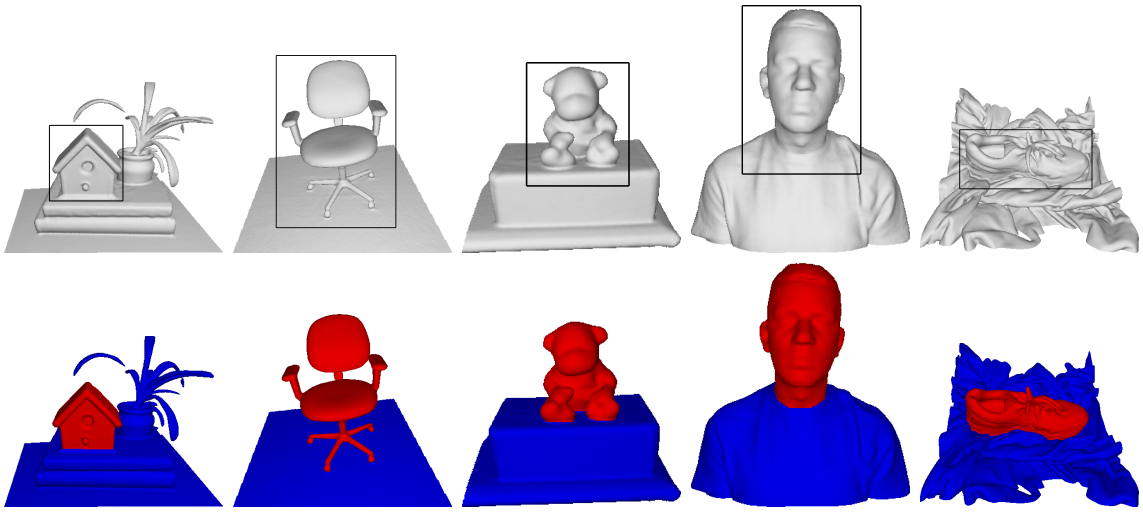

Figure 7: The results of our proposed mesh segmentation technique. The top row shows the reconstructed 3D scene with user input, and the bottom row shows our resulting segmentation.

bounding box does not completely contain another object, our system will converge to the desired segmentation.

The runtime of our system is adequate for interactive mesh segmentation. On commodity hardware, our method can segment a triangular mesh with several hundred thousand triangular faces in less than a second. For example, the 3D scene in Figure 1 contains 362,616 faces, and the cup is extracted in 0.981 seconds on an Intel Core i7 CPU.

4. Concluding remarks

In this paper, we presented 3D GrabCut, a novel method for interactively extracting a 3D object from a reconstructed 3D scene with a minimal amount of effort. In the near future, everyone will have the capability to construct a 3D model of their environment by utilizing depth sensing technology in their mobile devices. Therefore, it is critical to have an intuitive user interface to enable non-experts to segment 3D scenes. Techniques that use the standard stroke-based user interface require the user to provide multiple strokes. Often, it is not immediately clear to the user where the strokes should be placed resulting in the user trying numerous placements to achieve a good segmentation. Our proposed method, simplifies the segmentation task by requiring the user to simply draw a rectangular box around the desired object.

References

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (2006), 181–193. 1

[BJ01] BOYKOV Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 1, pp. 105–112 vol.1. 3

[FL*11] FAN L., LIU K., ET AL.: Paint mesh cutting. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 603–612. 1

[GF08] GOLOVINSKIY A., FUNKHOUSER T.: Randomized cuts for 3d mesh analysis. In *ACM Transactions on Graphics (TOG)* (2008), vol. 27, ACM, p. 145. 1

[Goo14] GOOGLE: Project tango. https://www.google.com/atap/projecttango/, 2014. 1

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P. S.: Hierarchical face clustering on polygonal surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), ACM, pp. 49–58. 1

[HR84] HOFFMAN D. D., RICHARDS W. A.: Parts of recognition. *Cognition 18*, 1 (1984), 65–96. 3, 4

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. In *Computer Graphics Forum* (2006), vol. 25, Wiley Online Library, pp. 283–291. 1, 2

[KLT05] KATZ S., LEIFMAN G., TAL A.: Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10 (2005), 649–658. 1

[KT03] KATZ S., TAL A.: *Hierarchical mesh decomposition using fuzzy clustering and cuts*, vol. 22. ACM, 2003. 1

[LHMR08] LAI Y.-K., HU S.-M., MARTIN R. R., ROSIN P. L.: Fast mesh segmentation using random walks. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling* (2008), ACM, pp. 183–191. 1, 3

[LZ04] LIU R., ZHANG H.: Segmentation of 3d meshes through spectral clustering. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (2004), IEEE, pp. 298–305. 1

[NDI*11] NEWCOMBE R. A., DAVISON A. J., IZADI S., KOHLI P., HILLIGES O., SHOTTON J., MOLYNEAUX D., HODGES S., KIM D., FITZGIBBON A.: Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and Augmented Reality*

(ISMAR), 2011 10th IEEE International Symposium on (2011), pp. 127–136. 1

[Occ14] OCCIPITAL, INC.: Structure. http://structure.io/, 2014. 1

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 309–314. 1, 3, 4

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. In *Computer graphics forum* (2008), vol. 27, Wiley Online Library, pp. 1539–1556. 1, 2, 3

[STK02] SHLAFMAN S., TAL A., KATZ S.: Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 219–228. 1, 3