Eurographics Workshop on Urban Data Modelling and Visualisation (2013)
V. Tourre and G. Besuievsky (Editors)

Online Reconstruction of Textured Triangle Meshes from Aerial Images

Tom Vierjahn[1,2,3], Jan Roters[2], Manuel Moser[1], Klaus Hinrichs[2], and Sina Mostafawy[1,3]

[1][rmh] new media GmbH, Cologne, Germany
[2]Visualization and Computer Graphics Research Group, Department of Computer Science, University of Muenster, Germany
[3]Department of Media, FH Duesseldorf University of Applied Sciences, Germany

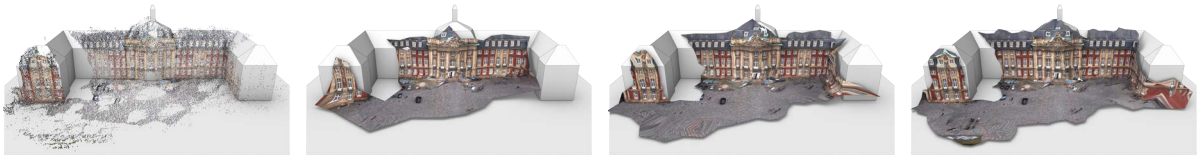


Figure 1: *Iterative reconstruction of a textured triangle mesh. From left to right: The point cloud created from a set of seven aerial images, and the triangle mesh after* 0.04 s, 0.2 s *and* 1 s *of reconstruction. The grey proxy is provided as a visual 3D-cue.*

Abstract

In this paper we present and evaluate a new online reconstruction algorithm to create a textured triangle mesh from a set of aerial images via an unorganized point cloud. Both the point cloud and the mesh are iteratively refined while allowing new aerial images to be added at any time during reconstruction. Texture coordinates are learnt to instantly visualize an initially rough approximation that gets refined as more data becomes available.
The new algorithm improves upon other systems that require the complete data to be acquired beforehand, and that apply offline, non-iterative reconstruction and processing. Thus, our algorithm is perfectly suited for time-critical applications, e. g., strategical visualization platforms for disaster and emergency response.

Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Texture I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Reconstructing 3D scenes from images is of major interest for several application domains, e. g., reverse engineering, cultural heritage and urban reconstruction. A remarkable work on the latter can be found in [FFGG*10], a survey of techniques in [MWA*12]. Most use offline reconstruction from previously acquired data, e. g., images.

The algorithm presented in this paper will be used in a strategic visualization pipeline [SFS*11]. Tight timing-restrictions make an iterative online approach indispensable: A rough approximation is reconstructed at first. The point cloud extracted from aerial images [RJ13] and the reconstructed surface [VHHM13] get refined as more images become available. In contrast to approaches like [GWO*10], texture information will be learnt during reconstruction.

Closely related to the surface reconstruction algorithm used in this paper are [AB10] and [RAB10]. A detailed discussion of other reconstruction algorithms can be found in [VHHM13]. However, they do not integrate textures.

2. Reconstruction Pipeline

The online reconstruction algorithm presented in this paper iteratively reconstructs a textured triangle mesh $\mathcal{M}$ from a

DOI: 10.2312/UDMV/UDMV13/001-004

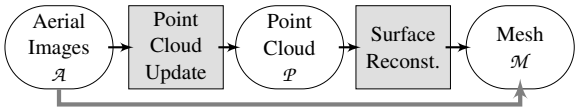


Figure 2: *Reconstruction pipeline: A set $\mathcal{A}$ of aerial images is used to reconstruct a set $\mathcal{P}$ of points from which a triangle mesh $\mathcal{M}$ is reconstructed, textured with the aerial images.*

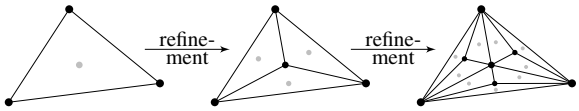


Figure 3: *Feature refinement: A 2D triangulation of feature points is repeatedly subdivided.*

growing set $\mathcal{A}$ of aerial images via an unorganized point cloud $\mathcal{P}$. Even though Fig. 2 might suggest sequential processing, *point cloud update* and *surface reconstruction* run in parallel: As new images are added to $\mathcal{A}$ *point cloud update* adds new points to $\mathcal{P}$ or updates existing points, while in the meantime *surface reconstruction* updates the mesh $\mathcal{M}$ according to previous updates to $\mathcal{P}$.

Rigorous mathematical notation will be used for a precise description. 3D position $\mathbf{p}$, 2D location $\mathbf{x}$ in an image A and texture coordinates $\mathbf{t}$ will be attributes assigned to either the points p, the vertices v or both. Maps $b_{\mathcal{S}} : e \in \mathcal{S} \mapsto b_{\mathcal{S}}(e)$ will be defined to assign an attribute b to an element e of a set $\mathcal{S}$ or to determine the attribute itself. The shorthands $b_e = b_{\mathcal{S}}(e)$ and $b = b_{\mathcal{S}}$ will be used for brevity if the exact meaning is clear from the context. Boldfaced symbols $\mathbf{b}$ denote vector-valued attributes and maps whereas script symbols $\mathcal{B}$ denote attributes and maps that provide sets.

2.1. Point Cloud Update

A point cloud $\mathcal{P}$ is created similar to the way presented in [FFGG*10]: 2D feature points are detected in aerial images and matched across pairs of images. The epipolar geometry is estimated and 3D points $\mathbf{p}_{p_s}$ are reconstructed based on Euclidean geometry. For each 3D point a sample p_s is added to $\mathcal{P}$ and a map $\mathbf{p}_{\mathcal{P}} : p_s \mapsto \mathbf{p}_{\mathcal{P}}(p_s) = \mathbf{p}_{p_s} \in \mathbb{R}^3$ is updated.

This paper proposes an online algorithm: Whenever new aerial images have been captured and registered, a sparse set of 3D points is reconstructed [RSH11] and added to the point cloud to provide an approximation. Afterwards, the point cloud is refined incrementally. Instead of searching for new feature points in the vicinity of other feature points, a 2D triangulation of the already detected feature points is repeatedly subdivided (cf. Fig. 3). The former technique would add redundant information to the point cloud, whereas the latter enhances entropy. New feature points are detected according to the subdivided triangulation. They are matched and new 3D points are extracted. That way, 2D feature points and corresponding 3D points can be extracted from newly acquired images up to a certain level of refinement, before the complete set of features in all images is further refined. Detail can be added where needed.

A detailed description of the above and an efficient technique to restrict the area in which to search for matching feature points is presented in an accompanying paper [RJ13].

2.2. Textured Surface Reconstruction

Artificial neural networks as presented in [AB10, RAB10, VHHM13] can be used to reconstruct a triangle mesh with arbitrary genus from the unorganized point cloud that was created in the previous step. All of these algorithms can be extended by the texture learning approach presented in this paper. However, *surface reconstructing Growing Neural Gas* (sGNG) [VHHM13] was used in this work. The basic learning algorithm that is similar in all of the above techniques will be briefly described here to derive *texture learning*.

2.2.1. Surface Reconstruction by Learning

sGNG and related algoritms implement an artificial neural network consisting of a set $\mathcal{V}$ of vertices v_i that are interconnected by edges $(v_j, v_k) \in \mathcal{E} \subseteq \mathcal{V}^2$ with $(v_j, v_k) \equiv (v_k, v_j)$. For surface reconstruction a 3D position $\mathbf{p}_{v_i}$ must be assigned to each vertex $v_i \in \mathcal{V}$ by a map $\mathbf{p}_{\mathcal{V}} : v_i \mapsto \mathbf{p}_{\mathcal{V}}(v_i) = \mathbf{p}_{v_i} \in \mathbb{R}^3$.

The 3D position $\mathbf{p}_{p_\xi}$ of a randomly selected point $p_\xi \in \mathcal{P}$ is presented as an input signal to the sGNG network in each learning iteration. The best matching vertex $v_b \in \mathcal{V}$ is determined with respect to the ℓ^2-norm $\|\cdot\|_2$:

$$v_b = \underset{v_i \in \mathcal{V}}{\arg\min} \|\mathbf{p}_{p_\xi} - \mathbf{p}_{\mathcal{V}}(v_i)\|_2$$

Afterwards v_b and all of its direct topological neighbours are moved towards the signal according to predefined learning rates β for v_b and γ for its neighbours. Let a map $\mathcal{N} : v_i \mapsto \mathcal{N}(v_i) = \mathcal{N}_{v_i} = \{v_n \mid (v_i, v_n) \in \mathcal{E}\}$ provide a set of all direct topological neighbours of v_i, then

$$\mathbf{p}_{v_b}^{(\vartheta+1)} = (1-\beta)\,\mathbf{p}_{v_b}^{(\vartheta)} + \beta\,\mathbf{p}_{p_\xi} \qquad (1)$$

$$\forall v_n \in \mathcal{N}_{v_b}^{(\vartheta)} \;:\; \mathbf{p}_{v_n}^{(\vartheta+1)} = (1-\gamma)\,\mathbf{p}_{v_n}^{(\vartheta)} + \gamma\,\mathbf{p}_{p_\xi} \qquad (2)$$

with superscript (ϑ) denoting the number of the current iteration. In general, to achieve good results, $\beta > \gamma$.

New vertices are added after certain numbers of iterations by splitting the longest edge $(v_h, v_q) \in \mathcal{E}$ incident to the most active vertex v_h, i. e., the vertex that was selected best match more often than the others:

$$v_q = \underset{v_i \in \mathcal{N}_{v_h}}{\arg\max} \|\mathbf{p}_{v_h} - \mathbf{p}_{\mathcal{V}}(v_i)\|_2$$

$$\mathcal{V}^{(\vartheta+1)} = \mathcal{V}^{(\vartheta)} \cup \{v_m\} \;,\; \mathbf{p}_{v_m} = \frac{\mathbf{p}_{v_h} + \mathbf{p}_{v_q}}{2} \qquad (3)$$

Finally, inactive vertices are removed from the network.

sGNG is the first algorithm of its kind that is able to create a complete set $\mathcal{F} \subseteq \mathcal{V}^3$ of triangular faces during learning without using post-processing. Therefore, a triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ is readily available at any time during surface reconstruction: An initially rough approximation that gets refined as more data becomes available.

A detailed description of the above, connectivity updates and triangle creation is presented in [VHHM13].

2.2.2. Texture Learning

Colour information can be easily extracted from the aerial images and assigned to the 3D points in a very dense point cloud. Learning vertex colours is a straightforward extension to reconstruction algorithms like sGNG. However, since colour is interpolated between the vertices during rendering, a very large number of small triangles has to be reconstructed to represent fine details. Therefore, sGNG is extended by *texture learning* to provide a highly detailed colour representation even on a coarse mesh.

Additional information about the points in $\mathcal{P}$ is needed to learn texture information. Let a map $\mathcal{A}_\mathcal{P} : p_s \mapsto \mathcal{A}_\mathcal{P}(p_s) = \mathcal{A}_{p_s} \subseteq \mathcal{A}$ provide the set $\mathcal{A}_{p_s}$ of aerial images that contain the feature points corresponding to $p_s \in \mathcal{P}$.

The 2D locations of feature points in the aerial images and thus the 2D locations of the corresponding p_s were determined in the previous step. This allows to define a map $\mathbf{x} : p_s, A_j \mapsto \mathbf{x}(p_s, A_j) = \mathbf{x}_{p_s,A_j} \in [0,1]^2 \subset \mathbb{R}^2$ that assigns such a 2D location $\mathbf{x}_{p_s,A_j}$ in $A_j \in \mathcal{A}_{p_s}$ to a point p_s.

Analogously to the point-to-image map $\mathcal{A}_\mathcal{P}$ let a map $\mathcal{A}_\mathcal{V} : v_i \mapsto \mathcal{A}_\mathcal{V}(v_i) = \mathcal{A}_{v_i} \subseteq \mathcal{A}$ provide the set of aerial images that is learnt by sGNG as a set of texture candidates for a vertex v_i. The images in $\mathcal{A}_{p_\xi}$ of an input signal p_ξ are in general used as new texture candidates for a vertex. However, texture candidates may become invalid as the vertex is moved during sGNG iterations.

To detect invalid texture candidates, a map $\tau : v_i, A_j \mapsto \tau(v_i, A_j) = \tau_{v_i,A_j} \in \mathbb{R}$ assigns a confidence measure τ_{v_i,A_j} to a vertex v_i that a given image A_j is suitable for texturing this vertex. This map is updated, whenever a signal provides texture candidates to a vertex:

$$\forall A_j \in \mathcal{A}_{v_i} \; : \; \tau^{(\vartheta+1)}_{v_i,A_j} = \begin{cases} \eta\, \tau^{(\vartheta)}_{v_i,A_j} & \text{, if } A_j \in \mathcal{A}_{p_\xi} \\ \mu\, \tau^{(\vartheta)}_{v_i,A_j} & \text{, otherwise} \end{cases}$$

with initially $\tau_{v_i,A_j} = 1$, and η slightly larger than 1, μ slightly smaller than 1.

Texture candidates are removed from $\mathcal{A}_{v_i}$ if they were not provided by p_ξ in the current iteration and if $\tau_{v_i,A_j} < \underline{\tau}$ with $\underline{\tau}$ denoting the minimal confidence allowed. Let $\underline{\mathcal{A}}_{v_i} = \left\{ A_j \mid A_j \in \mathcal{A}_{v_i} \wedge \tau_{v_i,A_j} < \underline{\tau} \right\}$, then

$$\mathcal{A}^{(\vartheta+1)}_{v_i} = \left(\mathcal{A}^{(\vartheta)}_{v_i} \cup \mathcal{A}_{p_\xi} \right) \setminus \underline{\mathcal{A}}_{v_i}$$

Let a map $\mathbf{t} : v_i, A_j \mapsto \mathbf{t}(v_i, A_j) = \mathbf{t}_{v_i,A_j} \in [0,1]^2 \subset \mathbb{R}^2$ assign a 2D texel location $\mathbf{t}_{v_i,A_j}$ in $A_j \in \mathcal{A}_{v_i}$ to v_i. Integrating learning of texture coordinates into sGNG is similar to the way positions are learnt (equations (1), (2) and (3)):

$$\forall A_j \in \mathcal{A}_{p_\xi} \; : \; \mathbf{t}^{(\vartheta+1)}_{v_b,A_j} = (1-\beta)\, \mathbf{t}^{(\vartheta)}_{v_b,A_j} + \beta\, \mathbf{x}_{p_\xi,A_j}$$

$$\forall A_j \in \mathcal{A}_{p_\xi} \; : \; \forall v_n \in \mathcal{N}^{(\vartheta)}_{v_b} \; : \; \mathbf{t}^{(\vartheta+1)}_{v_n,A_j} = (1-\gamma)\, \mathbf{t}^{(\vartheta)}_{v_n,A_j} + \gamma\, \mathbf{x}_{p_\xi,A_j}$$

$$\forall A_j \in \mathcal{A}_{v_m} \; : \; \mathbf{t}_{v_m,A_j} = \frac{\mathbf{t}_{v_h,A_j} + \mathbf{t}_{v_q,A_j}}{2} \; , \; \tau_{v_m,A_j} = \frac{\tau_{v_h,A_j} + \tau_{v_q,A_j}}{2}$$

with $\mathcal{A}_{v_m} = \mathcal{A}_{v_h} \cap \mathcal{A}_{v_q}$ and initialization $\mathbf{t}^{(\vartheta)}_{v_i,A_j} = \mathbf{x}_{p_\xi,A_j}$.

A single image has to be selected from the candidates to correctly render a textured triangle $\triangle v_k, v_l, v_m$. Let $\mathbf{d}_{\triangle,C} : A_j \mapsto \mathbf{d}_{\triangle,C}(A_j) \in \mathbb{R}^3$ provide the vector from the barycenter of the triangle to the camera position from where an aerial image A_j was taken. Let furthermore $\mathbf{g}_C : A_j \mapsto \mathbf{g}_C(A_j) \in \mathbb{R}^3$ provide the gaze direction of that camera. Then, a function $f_{\triangle,C} : A_j \to f_{\triangle,C}(A_j) \in [-1,1] \subset \mathbb{R}$ provides an orthogonality measure similar to the *form factor* that is used in *Radiosity* where

$$f_{\triangle,C}(A_j) = \frac{\left(\mathbf{n}^T_\triangle \cdot \mathbf{d}_{\triangle,C}(A_j)\right) \cdot \left(\mathbf{g}^T_C \cdot (A_j)\, \mathbf{d}_{\triangle,C}(A_j)\right)}{\|\mathbf{n}_\triangle\|_2 \cdot \|\mathbf{g}_C(A_j)\|_2 \cdot \|\mathbf{d}_{\triangle,C}(A_j)\|_2^2}$$

$$= \cos\left(\angle \mathbf{n}_\triangle, \mathbf{d}_{\triangle,C}(A_j)\right) \cdot \cos\left(\angle \mathbf{g}_C(A_j), \mathbf{d}_{\triangle,C}(A_j)\right)$$

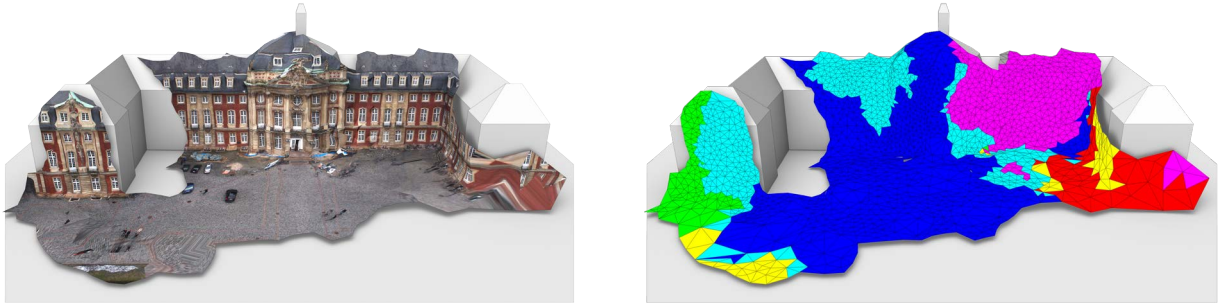
with $\mathbf{n}_\triangle$ denoting the normal of the triangle. Let $\tau_\triangle : A_j \to \tau_\triangle(A_j) = \tau(v_k, A_j) \cdot \tau(v_l, A_j) \cdot \tau(v_m, A_j)$ provide the product of confidence measures of the vertices of the triangle. Let finally $\mathcal{A}_\triangle = \mathcal{A}_{v_k} \cap \mathcal{A}_{v_l} \cap \mathcal{A}_{v_m}$ denote the set of common aerial images of the vertices of the triangle, then an image $A_\triangle$ will be used for texturing where

$$A_\triangle = \underset{A_j \in \mathcal{A}_\triangle}{\arg\max} \left(\tau_\triangle(A_j) \cdot f_{\triangle,C}(A_j) \right) \tag{4}$$

3. Results

The technique proposed in this paper was evaluated by reconstructing textured triangle meshes from unorganized point clouds that were extracted from a set of aerial images. Point cloud creation and sGNG surface reconstruction itself will not be evaluated in this paper. Detailed results can be found in [RJ13] and [VHHM13].

The pipeline was distributed among two computers: One reconstructed the points in Matlab using CPU and GPU, the other reconstructed the textured triangle mesh using sGNG on a single core of the CPU. A set of seven aerial images was used for the textured reconstruction examples in this paper. The images were taken with a camera mounted to a miniature unmanned aerial vehicle. Creating an initially sparse point cloud with 5,700 points took about 30 s. Additional 60,000 points were extracted in ten iterations of refinement in about 150 s. sGNG used this refined point cloud as an input for the examples in this paper.



(a) (b)

Figure 4: *Textured and colored meshes: (a) A textured mesh with 4,100 triangles, reconstructed in* 1 s. *The grey proxy is provided as a visual 3D-cue. (b) Color-coded sub-meshes. Each is textured with only one aerial image.*

Fig. 1 presents an overview of the point cloud (left) and a sequence of sGNG iterations: Instantly, after 0.04 s, a rough but fully textured approximation with 200 triangles is available for visualization. The triangle mesh is continuously refined to 1,000 triangles after 0.2 s, and up to 4,100 triangles after 1 s. An enlarged version is presented in Fig. 4 (a), providing fine visual detail on an approximation of the geometry. The triangle mesh is segmented into sub-meshes by evaluating equation (4). This segmentation is presented in Fig. 4 (b). Each sub-mesh is textured with only a single aerial image. The textures match well with little to no visual artifacts at the boundaries of the sub-meshes. However, slight distortions occur in areas where the point cloud was very noisy. The distortions and small sub-meshes on the right wing of the castle are due to very sparse sample points in that area.

4. Conclusion and Future Work

With the technique presented in this paper, a fully textured triangle mesh is created at virtually the same moment that a 3D point cloud has been extracted from a set of aerial images. Hence, detailed, close-up display is feasible even in time-critical applications, and a wealth of rendering and visualization techniques can be applied.

Segmentation into sub-meshes is advantageous. They enable efficient rendering. Using them for spatial subdivisions, local refinement and improved texture management is conceivable and will be addressed in future work. However, segmentation can lead to artifacts as noted, e. g., in [GWO*10]. We will evaluate their techniques to improve our pipeline. It was demonstrated in [VHHM13] that sGNG robustly handles even extremely large datasets. We are thus planning to extend the texture learning algorithm accordingly.

Acknowledgements

The project AVIGLE is part of the Hightech.NRW initiative funded by the Ministry of Innovation, Science and Research of the German State of North Rhine-Westphalia.

References

[AB10] ANNUTH H., BOHN C.-A.: Surface reconstruction with smart growing cells. In *Intelligent Computer Graphics 2010*, Plemenos D., Miaoulis G., (Eds.), vol. 321 of *Studies in Computational Intelligence*. Springer Berlin / Heidelberg, 2010, pp. 47–66. 1, 2

[FFGG*10] FRAHM J.-M., FITE-GEORGEL P., GALLUP D., JOHNSON T., RAGURAM R., WU C., JEN Y.-H., DUNN E., CLIPP B., LAZEBNIK S., POLLEFEYS M.: Building rome on a cloudless day. In *Proc. 11th European Conf. Computer vision: Part IV* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 368–381. 1, 2

[GWO*10] GAL R., WEXLER Y., OFEK E., HOPPE H., COHEN-OR D.: Seamless montage for texturing models. *Comput. Graph. Forum 29*, 2 (2010), 479–486. 1, 4

[MWA*12] MUSIALSKI P., WONKA P., ALIAGA D. G., WIMMER M., VAN GOOL L., PURGATHOFER W.: A survey of urban reconstruction. In *EG 2012 - State of the Art Reports* (Cagliari, Sardinia, Italy, 2012), Cani M.-P., Ganovelli F., (Eds.), Eurographics Association, pp. 1–28. 1

[RAB10] RÊGO R. L. M. E., ARAÚJO A. F. R., BUARQUE DE LIMA NETO F.: Growing self-reconstruction maps. *IEEE Trans. Neural Netw. 21*, 2 (Feb 2010), 211–223. 1, 2

[RJ13] ROTERS J., JIANG X.: Incremental dense reconstruction from sparse 3d points with an integrated level-of-detail concept. In *Advances in Depth Image Analysis and Applications* (2013), Lecture Notes in Computer Science, Springer Berlin Heidelberg. to appear. 1, 2, 3

[RSH11] ROTERS J., STEINICKE F., HINRICHS K. H.: Quasi-real-time 3d reconstruction from low-altitude aerial images. In *Proc. 40th Urban Data Management Society Symp.* (Sep 2011), Zlatanova S., Ledoux H., Fendel E., Rumor M., (Eds.), vol. 40 of *UDMS Annual 2011*, CRC Press/Balkema, pp. 231–241. 2

[SFS*11] STROTHOFF S., FELDMANN D., STEINICKE F., VIERJAHN T., MOSTAFAWY S.: Interactive generation of virtual environments using muavs. In *Proc. Int'l Symp. VR innovation (ISVRI)* (2011), pp. 89 – 96. 1

[VHHM13] VIERJAHN T., HENRICH N., HINRICHS K., MOSTAFAWY S.: *sGNG: Online Surface Reconstruction based on Growing Neural Gas*. Tech. rep., Dept. Computer Science, Univ. Muenster, Germany, 2013. 1, 2, 3, 4