Computers & Graphics 43 (2014) 11–20

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

A clustering-based method to estimate saliency in 3D animated meshes ☆

Abdullah Bulbul [a,b,*], Sami Arpa [b,c], Tolga Capin [d]

[a] Vision Science Graduate Group, University of California at Berkeley, Martin Banks' lab, 360 Minor Hall, Berkeley, CA 94720, USA
[b] Department of Computer Engineering, Bilkent University, Turkey
[c] School of Computer and Communication Sciences, EPFL, Switzerland
[d] Department of Computer Engineering, TED University, Turkey

ARTICLE INFO

Article history:
Received 2 October 2013
Received in revised form
18 April 2014
Accepted 29 April 2014
Available online 16 June 2014

Keywords:
Computer graphics
Visual perception
Saliency
Animated meshes

ABSTRACT

We present a model to determine the perceptually significant elements in animated 3D scenes using a motion-saliency method. Our model clusters vertices with similar motion-related behaviors. To find these similarities, for each frame of an animated mesh sequence, vertices' motion properties are analyzed and clustered using a Gestalt approach. Each cluster is analyzed as a single unit and representative vertices of each cluster are used to extract the motion-saliency values of each group. We evaluate our method by performing an eye-tracker-based user study in which we analyze observers' reactions to vertices with high and low saliencies. The experiment results verify that our proposed model correctly detects the regions of interest in each frame of an animated mesh.

1. Introduction

The human visual system (HVS) frequently shifts focal attention to the subsets of a scene, that is, to salient feature points. The visual acuity and the details transferred from the real world to the HVS change with these shifts. Fixation movements are the most important movements of the eye; the fixation mechanism allows us to direct the eyes towards objects of interest. While this process is automatically employed by the HVS, saliency detection mechanisms are not yet fully understood. For 3D graphics, automatic detection of salient features may provide significant advances in various problems, including selective rendering, view-point selection, retargeting, symmetry detection, segmentation, and 3D model compression.

Many view-independent saliency detection models have been proposed for 3D scenes in the graphics literature. In these methods, geometric features such as mean curvature differences at different scales and average variations between two polygons are considered, but temporal variations of the geometry are not well integrated. Regarding this drawback, we use HVS mechanisms supported by motion-related psychophysical experiments to develop a metric calculating the saliency of 3D objects based on their motion. Current research shows that while motion by itself does not attract attention, its attributes, such as initiation, may make it more salient. This paper presents a saliency model based on the effect of motion states on the attractiveness level of a visual stimulus.

The main contribution of this paper is a new approach to determine perceptually significant elements in animated 3D scenes using a motion-saliency model. Our proposed approach is based on clustering vertices with similar behaviors. To cluster vertices in each frame of a deforming mesh sequence are analyzed according to their motion properties. Vertices with similar motion behaviors are perceptually grouped with a Gestalt approach, thus each cluster is analyzed as a single unit. Representative vertices for each cluster are therefore used to extract the motion-saliency values of their clusters. To evaluate our model, we performed a user study to analyze observers' reactions to objects with high and low saliency values. The results of the experiment verify that the proposed metric correctly identifies the mesh regions with high motion saliency.

The paper is organized as follows: In Section 2, we present a review of previous studies in computer graphics utilizing the motion perception principles and the psychological principles that inform our method. Section 3 presents our 3D cluster-based motion-saliency estimation method. Section 4 presents the user study and its results. Section 5 presents a discussion and Section 6 concludes.

2. Related work

2.1. Concepts in visual attention and saliency

The visual attention mechanism can be divided into two components: bottom-up and top-down attention.

☆This article was recommended for publication by M. Spagnuolo.

* Corresponding author at: Vision Science Graduate Group, University of California at Berkeley, Martin Banks' lab, 360 Minor Hall, Berkeley, CA 94720, USA.
Tel.: +1 510 642 7679, fax: +1 510 643 5109.

http://dx.doi.org/10.1016/j.cag.2014.04.003
0097-8493/

Bottom-up attention: The bottom-up component of visual attention is driven merely by the properties of the visual scene, regardless of the viewer's intention. Viewer-independent factors (irrespective of personal tasks, experiences, etc.) direct the visual attention and are part of the bottom-up component.

Saliency, a bottom-up property, is mainly related to the differences of an object's various visual properties and its surroundings. The neurons employed in the visual system respond to image differences between a small central region and a larger surrounding region [1], in a process called the center-surround mechanism. Through this mechanism, the differences between a property and its surroundings stimulate the visual system. If an object is notably different from its surroundings, it becomes salient. This difference can be in terms of one or more properties of the object, such as hue, luminance, orientation or motion. A highly salient object pops out from the image and immediately attracts attention. This process is unconscious and operates faster than top-down, or task-oriented attention. The speed of bottom-up attention is usually between 25 to 50 ms per item, while task-oriented top-down attention takes more than 200 ms [1].

Top-down attention: What are we looking for greatly affects our visual perception. When we look for a specific type of object or for a specific property we may perceive many details that we may not perceive in a casual glance. On the other hand, biasing perception towards a specific target can make other objects less perceivable [2].

After objects have been selected from the scene in a bottom-up fashion, goal-oriented top-down attention determines *what* is perceived. This phase of attention includes constraining the recognized scene based on scene understanding and object recognition [1]. When a scene is constrained by the visual system, the region that gets the most attention is promoted, which is known as the winner-take-all principle [1].

With a search task to browse a scene, the HVS is optimally tuned according to the search goal, such that the features of the target become easily recognizable [3]. Interestingly, our visual system is not adjusted to the exact features of a search target, but adjusted to differentiate these features in the optimal way. For example, if our goal is to find a slightly right slanting object among objects oriented in an upwards direction, our sensitivity is tuned to that exaggerated feature in the target object to simplify differentiation. Similarly, when our attention is tuned to a search goal, we may not notice objects unrelated to our task even if they are easily visible; this phenomenon is called inattentional blindness [4].

Inhibition of return: Another principle of visual attention is called inhibition of return, first described in 1984 by Posner and Cohen [5], and which allows our visual system to perceive an entire scene rather than focusing only on the visually most attractive region. According to this principle, when a region is attended to once, our perception of that region is inhibited after the first 0.3 s and object recognition in that location decreases over approximately 0.9 s. As a result, our attention moves to a new region, enabling a search of different and novel regions on the visual periphery.

Motion perception: A difference of position in our visual field results in a sense of motion; this process requires a temporal analysis of the contents in our visual field. When two different images fall into our retina sequentially, our visual system must identify whether those images represent the same object in different positions or whether they are different objects. If the HVS determines that it is the former case, it has established that the object is moving. The HVS can easily perceive objects as smoothly moving, but the mechanism that detects motion is not that simple. Working out spatial relations is easier than solving temporal relations [6].

A proposed model to explain motion detection in the HVS is Reichardt's motion detector [7]. This device is based on small units responsible for detecting motions in specified directions. These units compare two retinal image points, and if the same signal appears in these two points with a small delay, the units detect motion in their specific direction [6]. Along with color, depth, and illumination, center-surround organization is also applied to motion processing in the HVS. The neurons processing motion have a double-opponent organization for direction selectivity [8], meaning that motion-detecting modules can inhibit their surroundings; motion must be differentiable compared to its surroundings to be detected.

In the spatial domain, the HVS tends to group stimuli by considering their similarities and proximity as introduced in the Gestalt principles. It is shown that the HVS also searches for similarities in the temporal domain and can group stimuli by considering their parallel motions [9]. According to this process, a group of moving dots with the same direction and speed could be perceived as a moving surface.

Visual motion may be referred to as salient because it has temporal frequency. On the other hand, recent studies in cognitive science and neuroscience have shown that motion by itself does not attract attention. However, phases of motion (e.g., motion onset, motion offset, continuous motion) have different degrees of influence on attention. Hence, each phase of motion should be analyzed independently. Abrams and Christ [10] experimented with different states of motion to observe the most salient one. They indicated that the onset of motion captures attention significantly compared to other states. Immediately after motion onset, the response to stimulus slows from the effect of the inhibition of return, and the attentional sensitivity to that stimulus is lost. Singletons, having a different motion than others within stimuli, capture attention in a bottom-up, stimulus-driven way. If there is a search target, only feature singletons attract attention. However, abrupt visual onsets capture attention even if they are not a target [11]. Hillstrom and Yantis [12] also showed that the appearance of new objects captures attention significantly compared to other motion cues and that motion offset and continuous motion do not capture the same level of attention.

2.2. Computational models of visual attention and saliency

Itti et al. [13,1] describe one of the earliest methods to compute the saliency of 2D images. To calculate the saliency of a region, they compute the Gaussian-weighted means of intensity, orientation, and color opponency properties in narrow and wide scales; the differences between these scales provide information on how a region is compared to its surroundings.

Lee et al. [14] introduced the concept of mesh saliency in 3D graphical models. In their work, the saliencies of mesh vertices are computed based on the mesh geometry. Their proposed mesh saliency metric is based on the center-surround operator on Gaussian-weighted mean curvatures. They use the computed saliency values to drive the simplification of 3D meshes, implementing Garland and Heckbert's Qslim method [15] for simplifying objects based on quadric error metrics.

The mesh saliency metric was improved by Liu et al. [16], who discuss two main disadvantages of Lee et al.'s work [14]. One disadvantage is that the Gaussian-weighted difference of fine and coarse scales can result in the same saliency values for two opposite and symmetric vertices because of the absolute difference in the equation. The other one is that combining saliency maps at different scales makes it difficult to control the number of critical points. Therefore, instead of the Gaussian filter, Liu et al. use a bilateral filter and define the saliency of a vertex as the Gaussian-weighted average of the scalar function difference between the neighboring vertices and the vertex itself. Kim et al. [17] presented a user study that compares the performance of the

previous mesh saliency methods by considering fixation locations for 3D rendered images and proposed a normalized saliency measure.

Leifman et al. [18] proposed a saliency-based viewpoint selection method that accounts for the distance to the foci of attention. Another saliency metric and measure for the degree of visibility is proposed by Feixas et al. [19], who use the Jensen–Shannon (JS) divergence of probability distributions by evaluating the average variation of JS divergence between two polygons, yielding similar results to Lee et al. [14]. A saliency map for selective rendering that uses colors, intensity, motion, depth, edges, and habituation (which refers to saliency reduction over time as the object stays on the screen) is developed using Graphics Processing Unit (GPU) [20]. That saliency map is based on the model suggested by Itti and Koch [21]. Recently, Chen et al. [22] investigated Schelling points on 3D surfaces, which were obtained from viewers as salient features and used for predicting other shapes by analyzing the prior user data.

3. Overview

Saliency calculations for 3D mesh models are generally performed for each vertex of a model [14,23], which can be quite expensive because models generally have a large number of vertices. Furthermore, when we look at a 3D mesh, we do not recognize each vertex as a separate part of an object; we rather group similar vertices perceptually and regard each group as a single unit, which enables a common analysis of vertices that move together.

Our proposed approach is based on clustering vertices that exhibit similar behaviors. Fig. 1 illustrates an overview of the proposed method. Each frame of a vertex-animated mesh sequence is analyzed to cluster the vertices according to their motion properties. Because vertices with similar motion behaviors are perceptually grouped with a Gestalt approach, each cluster is analyzed as a single unit after the clustering process. Thus, representative vertices for each cluster are used to extract the motion saliencies of their clusters. Since the number of items to be analyzed reduces significantly (from the total number of vertices to the number of clusters) via the clustering process, the saliency calculation can be performed in real time.

3.1. Motion saliency model

By considering the psychological literature given in Section 2, and through analyzing psychophysical experiments [24], we describe a model to compare the attention values of objects in terms of motion. While we know that motion by itself does not attract attention, its attributes, such as initiation, may do so. The proposed model is based on the effect that motion states have on the attractiveness level of a visual stimulus.

First, we consider motion attributes to discriminate between different states of an object's motion. Six states (explained in Table 1) form the essence of a motion cycle (Fig. 2).

In a 3D scene, the motion-saliency model calculates instant saliency values moment by moment. The motion state for each object is detected and the corresponding saliency values are calculated as time dependent variables, as shown in Table 2. An initial saliency value is assigned to each motion state, considering the dominancy of the states among each other; these values decay over time. When a state change occurs, the attention to this region remains for approximately 0.3 s and disappears in 0.9 s, according to the inhibition-of-return mechanism, as explained in Section 2. Thus, when a state change is identified, its effect is calculated for the following frames according to the formulas in Table 2. When a new state change occurs before the effect of the previous motion state vanishes, an object is affected by two motion states at the same time. In such cases, the maximum saliency value is selected as the final saliency value.

Table 1
States of motion.

Static	No change of location
Object appearance	Appearance of an object, which was not previously present, on the screen
Motion onset	Start of a motion (static to dynamic)
Motion offset	End of a motion (dynamic to static)
Continuous motion	State of keeping the motion at the same velocity
Motion change	Change in the direction or speed of a motion

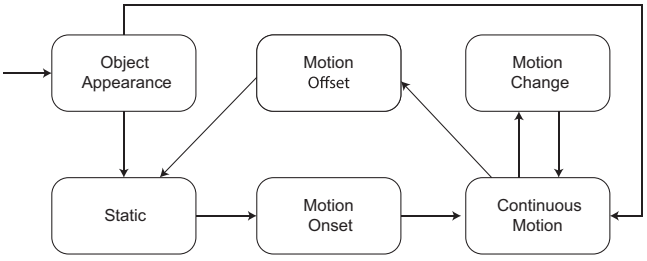

Fig. 2. Motion cycle of perceptually distinct motion states.

Table 2
Motion-saliency model.

States	$t \leq 300$ ms	300 ms $< t <$ 900 ms	900 ms $\leq t$
STATIC	$1k$	$1k$	$1k$
ONSET	$10k$	$10k\ ((4.2-4t)/3s)$	$2k$
OFFSET	$2k$	$2k((7.5-5t)/6s)$	$1k$
CONTINUOUS	$2k$	$2k$	$2k$
CHANGE	$10k$	$10k((4.2-4t)/3s)$	$2k$
APPEARANCE	$10k$	$10k((2.9-3t)/2s)$	$1k$

Individual attention values for changing motion states. (From [24], ©2011 Springer, reprinted with permission. For each motion state an initial saliency value is assigned considering its dominancy among the others. The dominancy of states in terms of saliency is obtained from motion-related psychophysical experiments. k is determined as a motion-saliency unit, discretized for computer space. Refer to Arpa et al. [24] for more details.).

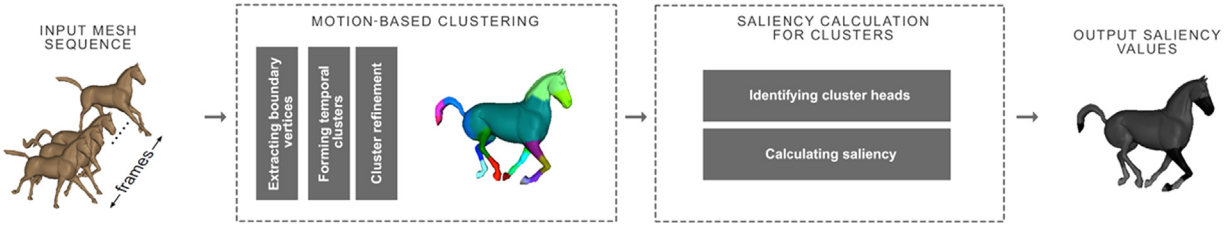

Fig. 1. Overview of the cluster-based saliency calculation.

3.2. Motion-based clustering

In this clustering technique, our aim is to group vertices of similar motions into the same cluster. Thus, the generated clusters should contain vertices that have very close motions throughout the animation. Additionally, we need to form clusters in which all vertices are connected. For this purpose, we utilize the velocities of vertices and their connectivity information. The clustering procedure is summarized in Algorithm 1:

Algorithm 1. Pseudocode for motion-based clustering.

```
for all frames of the animation do
    Find differential velocities of all vertices
    Extract boundary vertices using differential velocities
    Assign vertices inside a bounded area to a separate cluster
end for
Refine clusters to get final clustering
```

Finding differential velocities: In this step, our aim is to find vertices that have different velocities than their surroundings. Since vertices in a group have similar velocities throughout the animation, we assume that vertices with high relative velocities to their neighbors reside in the boundaries between clusters. We call the difference between the velocity of a vertex v and the Gaussian-weighted mean velocity in its surrounding region as the *differential velocity* of vertex v (denoted by $\overrightarrow{dv(v)}$), and calculate it as follows:

$$\overrightarrow{dv(v)} = |\overrightarrow{vel(v)} - \overrightarrow{G(vel, s, v)}|, \tag{1}$$

where $\overrightarrow{vel(v)}$ is the velocity of v and $\overrightarrow{G(vel, s, v)}$ is the Gaussian-weighted mean velocity of the surround s of v. For s, we use 0.036 of the size of the mesh's bounding box as the radius of the surrounding region, similar to Lee et al. [14]. We calculate the differential velocity calculations on a vector basis instead of considering scalar values so that we can differentiate velocities in different directions.

Fig. 3 illustrates differential velocities on a 3D model. As shown, differential velocities are higher (shown in yellow) in the boundary regions those separate the concrete regions, which move together.

Fig. 3. Differential velocities in a 3D model. Yellow regions express high differential velocities. The figure shows the absolute amounts of differential velocities in a scalar manner for a better presentation. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Extracting boundary vertices: After calculating the differential velocities of vertices, we select vertices with higher differential velocities to be the boundary vertices. The set of boundary vertices (BV) is composed of the vertices having higher differential velocities than a threshold:

$$BV = v \in V || dv(v)| > t_{bound}, \tag{2}$$

where V is the set of all vertices in the mesh and t_{bound} is the lower threshold for the absolute differential velocities of boundary vertices. The vertices in BV will be used to form clusters; thus, having more boundaries results in more clusters. Selecting a high t_{bound} value results in fewer boundary vertices and fewer clusters. Having fewer clusters is worse than having redundant clusters because the former results in incorrect saliency calculations; the latter is safe but results in computational overhead. Therefore, after experimenting with different threshold values, we selected a low value for t_{bound}: 0.001 percent of the maximum absolute differential velocity among all vertices of the mesh. Furthermore, most animated meshes have a skeletal structure with rigid body parts (e.g., foot and head); while vertices belonging to a rigid part have almost identical motion properties and very low differential velocities, vertices connecting these rigid portions usually have much higher differential velocities, making boundary extraction much easier. For such meshes, selecting a different t_{bound} makes little difference: t_{bound} value of 0.01 and 0.0001 resulted in the same final clusterings for a hand model [25].

Forming temporal clusters: In each frame of the animation, boundary vertices are used to form clusters, and the generated clusters are merged with the clusters from the previous frame. In this way, clusters are accumulated throughout the animation and each motion in any frame forms a cluster in the final clustering. The pseudocode to form the clusters in a single frame is shown in Algorithm 2:

Algorithm 2. Generating temporal clusters at each frame.

```
initialize NonClusteredVertices with all vertices except
    boundaries
initialize currentCluster with 1
while NonClusteredVertices is not empty do
    initialize TraceList
    vertex v=first element in NonClusteredVertices
    assign v to currentCluster
    remove v from NonClusteredVertices
    add v to TraceList
    while TraceList is not Empty do
        for all neighbors n of v do
            if n is not boundary then
                assign n to currentCluster
                add n to TraceList
                remove n from NonClusteredVertices
            end if
        end for
        remove v from TraceList
        v=first element of TraceList
    end while
    increment currentCluster
end while
```

Fig. 4 presents samples of temporal clusters throughout an animation. As shown, the boundary vertices are not assigned to any cluster in this phase because they will be assigned to the closest clusters to them in the cluster refinement phase.

Fig. 4. From left to right (except the rightmost): clustering progression throughout an animation. The leftmost image is the initial state, with a single cluster. As new frames are analyzed, new clusters are formed accumulatively (white regions depict the boundary vertices and other colors show generated clusters up to that frame). The rightmost image shows the final clustering after clustering refinement phase. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

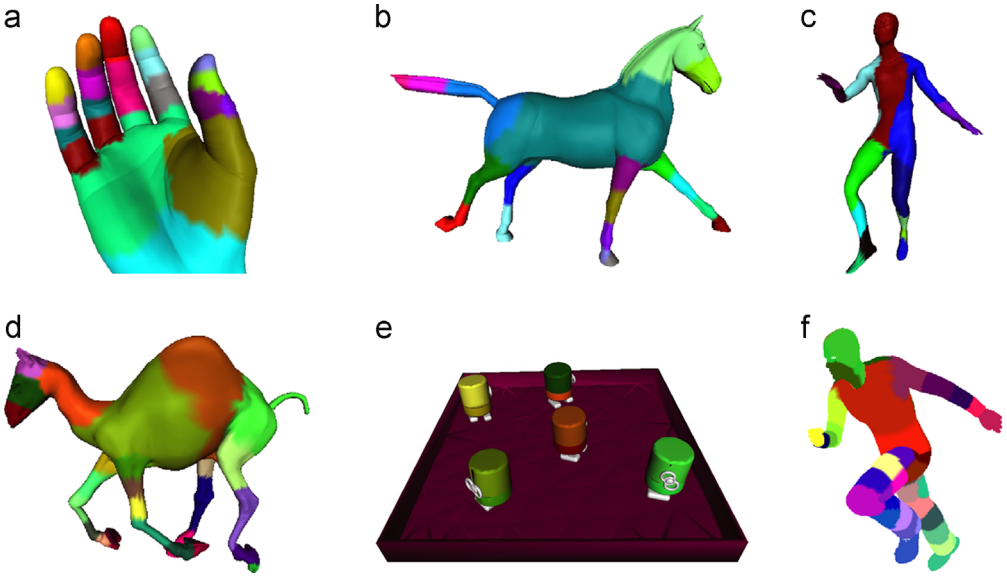

Fig. 5. Clustering results for several 3D models. The number of generated clusters from (a) to (f) is 21, 21, 14, 28, 12, and 46.

Algorithm 3. Cluster refinement.

while There is an update in last iteration **do**
  for all $e \in Edges$ **do**
    v 1 and v 2 are the vertices connected by e
    if $v1.cluster \neq v2.cluster$ **then**
      $d_{V1toC1} = dist(v1, v1.cluster)$
      $d_{V1toC2} = dist(v1, v2.cluster)$
      $d_{V2toC1} = dist(v2, v1.cluster)$
      $d_{V2toC2} = dist(v2, v2.cluster)$
      if $d_{V1toC1} > d_{V1toC2}$ **then**
        assign v 1 to v 2.*cluster*
      end if
      if $d_{V2toC1} < d_{V2toC2}$ **then**
        assign v 2 to v 1.*cluster*
      end if
    end if
  end for
end while

Cluster refinement: After temporal clusters are set for all frames, the clusters are refined to form the final clustering. In this phase, each vertex is assigned to the cluster with the most similar motion behavior to that vertex throughout the animation. The distance of a vertex v to a cluster C is calculated by summing up the squared differences between the velocity of v and the mean velocity of vertices belonging to C in each frame. This distance calculation is performed as follows:

$$dist(v, C) = \sum_{f=1}^{F} \left| \overrightarrow{vel(v,f)} - \frac{\sum_{w \in C} \overrightarrow{vel(w,f)}}{|C|} \right|^2, \quad (3)$$

where F is the number of frames throughout the animation and $\overrightarrow{vel(v,f)}$ is the velocity of vertex v in frame f. Algorithm 3 is performed to finalize the clustering process. Several clustering results are presented in Fig. 5.

The HVS groups similar motion behaviors, and therefore, each part of an animated mesh with a different motion characteristic forms a perceptible unit. This requires having more clusters for a model with a larger number of distinguishable motion behaviors. As indicated in Fig. 5, the number of clusters for the presented 3D models changes between 12 and 46. Our model generates more clusters for an animated mesh with a larger number of distinct motion properties, which is a desired outcome.

3.3. Saliency calculation for clusters

Identifying cluster heads. In this process, each cluster is handled as an object. For each cluster, we first select a representative vertex, called a cluster head. Then, the motions of the cluster

heads are analyzed throughout the animation to calculate the saliency of their clusters.

All the vertices in a cluster have similar motion properties, which is the reason why they are in the same cluster; however, it is best to select a cluster head whose motion properties resemble the entire cluster as much as possible. Therefore, we select as the cluster head as the vertex with the smallest distance from the cluster centroid, according to the distances given in Eq (3). By cluster centroid, we mean an imaginary vertex having the average velocity of all vertices in a cluster in each frame of the animation. Such a selection strategy computationally minimizes the squared error between the cluster head and the overall group. Since the number of cluster heads is much less than the number of vertices, calculating saliency on them could be done in real time.

Calculating saliency: We use the motion-saliency model from [24] to assign a saliency value for each cluster, as described in Section 3.1. Once the motion state of a cluster is detected, its saliency value is calculated as a time-dependent variable. We define six motion states (see Table 2), each of which triggers a different perceptual response.

The motion state of a cluster head is extracted using its velocity relative to the whole object. Motion caused by the movement of the entire object should be avoided in saliency calculations for clusters; thus, we subtract the average velocity of all clusters from the velocities of the cluster heads to obtain their relative velocities, as follows:

$$\overrightarrow{rv(v,f)} = \overrightarrow{vel(v,f)} - \frac{\sum_{w \in V} \overrightarrow{vel(w,f)}}{|V|}, \tag{4}$$

where V is the set of all vertices and f is a frame of the animation sequence.

The relative velocities of the cluster heads and two threshold values, t_{onset} and t_{change}, are used to extract the motion states, calculated with the following conditions:

$$State(v) = \begin{cases} |rv(v,f)| > t_{onset} \& |rv(v,f-1)| < t_{onset} \Rightarrow MotionOnset \\ |rv(v,f)| < t_{onset} \& |rv(v,f-1)| > t_{onset} \Rightarrow MotionOffset \\ |rv(v,f) - rv(v,f-1)| > t_{change} \Rightarrow MotionChange \\ |rv(v,f)| > t_{onset} \Rightarrow Continuous \\ otherwise \qquad Static \end{cases}$$

The HVS is not good at recognizing speed change [26], which is why we use a different threshold for motion change in our algorithm. In general, we set t_{change} as $2*t_{onset}$ and t_{onset}, as follows:

$$t_{onset} = \tan 0.15^{\circ} * d_{user} * k_{wtoc}, \tag{5}$$

where 0.15° is the drift velocity of the eye, which can be traced like a static situation [27], d_{user} is the distance from the user (or the camera) to the model, and k_{wtoc} is a constant to convert the result from world space to computer space. t_{onset} is the approximated 3D velocity in model space, which corresponds to a retinal velocity of 0.15°/s.

Each state is related to a saliency value, which is identical to the corresponding value shown in Table 2. Saliency values are assigned to vertices according to this table based on their motion states. Furthermore, according to our observations, a strong motion onset (starting with a higher velocity) attracts more attention, so we weight the saliencies according to the relative velocities of the cluster heads when a motion onset or motion change occurs. Eq. (6) shows this operation:

$$saliency(v)' = \frac{(saliency(v) - s_{base}) * |rv(v)|}{rv_{max}}, \tag{6}$$

where rv_{max} is the highest relative velocity at the moment and s_{base} is the saliency value given to continuously moving vertices.

After the saliency values for cluster heads have been calculated, they are spread to all vertices in their clusters. Fig. 6 shows the calculated saliencies for several 3D models. In our implementation, after the clustering phase (which could be done as a preprocess), the average saliency calculation time for a frame of the presented models was 0.81 ms on a 2.6 GHz Intel Core i7 processor. This corresponds to only 4.05% of the allocated time to render a frame of an animation, which runs at 50 frames per second, and enables calculating saliencies on the fly, without considerably decreasing the rendering performance.

For saliency calculations, we use velocities in 3D space instead of 2D velocities projected to the screen space because the HVS can extract absolute 3D velocity from 2D retinal images [6]. However, because the proposed saliency calculation method can work in real-time, the retinal velocities of the cluster heads could also be calculated and used for saliency calculations. This approach would not significantly affect the saliency calculation time because it only requires projecting a cluster's 3D velocity to the retinal image space by a simple matrix multiplication, which is already performed for each vertex of a model at each frame.

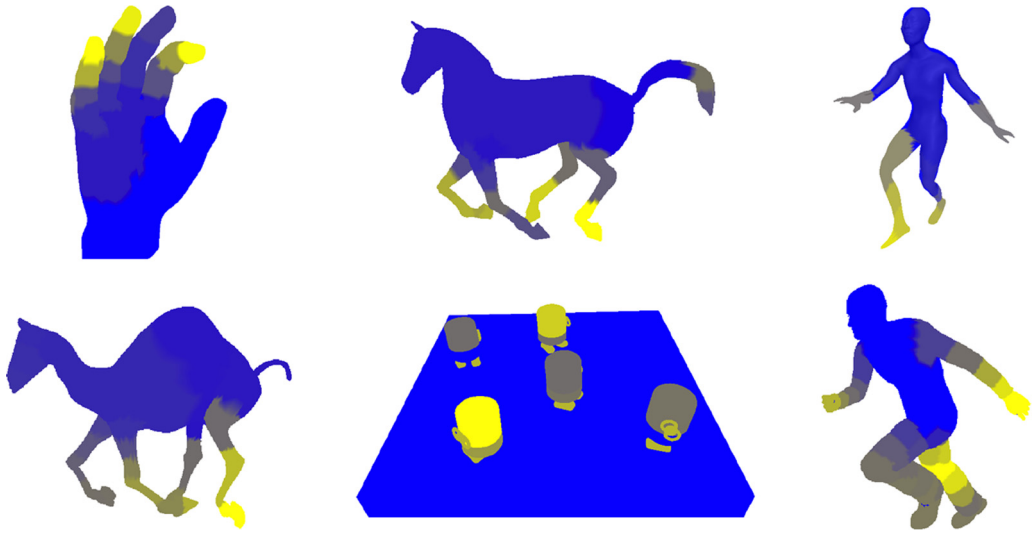

Fig. 6. Calculated saliencies for 3D models. Yellow regions show more salient parts of the models while blue depicts low saliency. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

4. Experiments

4.1. Experiment design

We performed an eye-tracker-based evaluation of our system. Ten graduate students with an average age of 24.6 with normal or corrected-to-normal vision participated in this study voluntarily. All subjects viewed stimuli shown on a 22 in LCD display from a distance of 70 cm and their gaze positions were recorded via an SMI RED eye tracker.

It is known that human attention is significantly affected by tasks and prior knowledge; Yarbus [2] clearly showed how human gaze is directed according to a task. We tried to avoid the task effect on gaze positions by telling the subjects to observe the scenes freely. Also well known is that the heads of natural objects attract significant attention, which is another important factor that could bias the results [28]. McDonnell et al. [29] showed that in a crowd model, head variety alone is sufficient to give the impression of agent variation. Thus, our user experiments did not include animated meshes with strong head features. Although one of the scenes (Fig. 7right) has a visible face, it covers a small area of the visual field and the presence of multiple objects decreases the dominant effect of the face on saliency distribution.

4.2. 3D Meshes used

The stimuli used in the experiment were short videos of three animated 3D meshes (Fig. 7) obtained from the Utah 3D Animation Repository [25]: Hand, Toasters, and Fairy. The models have around 16 000, 11 000, and 170 000 triangular faces, respectively. More information can be found at http://www.sci.utah.edu/~wald/animrep/.

4.3. Analysis method

To analyze the saliency values of gaze points we rendered the 3D meshes according to their values such that the brightest part is the most salient. In this way, we obtained 2D saliency images and measured the Kullback–Leibler (KL) divergence between the saliency values of human fixation points and the saliency values of 500 randomly selected fixation points.

Kullback–Leibler divergence is a non-symmetric indicator of how a probability distribution differs from another probability distribution and is widely used in image-based saliency analysis research [30–32]. In our case, calculated saliency values of human gaze points form the discrete distribution P_h, and calculated saliency values of random points form the discrete distribution P_r. A higher KL divergence of P_h from P_r points to a greater distinction from randomness and indicates a better saliency estimation. The KL divergence between P_h and P_r is calculated as follows:

$$KL(P_h, P_r) = \sum_i P_h(i) \ln\left(\frac{P_h(i)}{P_r(i)}\right). \quad (7)$$

Some of the stimuli we used in the experiment contain a plain background, which does not attract any attention. Having random fixations on the background could result in misleadingly high KL divergence values. So we omitted such points.

In our analysis, because of the high acuity around the gaze point and the inaccuracy of the eye tracker, we took into consideration approximately 1.5° of visual field around the designated fixation point, which is equivalent to blurring the saliency images, and makes the results closer to the random case.

Eye movements consist of two main categories of events, which occur sequentially. Fixations occur when observing a specific region in the visual field, and are for perceiving the visual information. The focus of attention then quickly switches to another point via a very rapid saccadic eye movement, which is then followed by another fixation. One can use either fixations or saccades in the analysis. We tried both for one of our stimuli and they gave similar results. We prefer using fixation points in our analysis, however, because fixations cover a longer period and provide more data to analyze.

4.4. Results

We compared the proposed method to three other studies: Lee et al. [14], Bulbul et al. [23], and Walther and Koch [33]. Lee et al.'s mesh saliency [14] approach is a curvature-based saliency estimation for static 3D meshes and Bulbul et al.'s method [23] additionally takes vertex motion properties into account in a center-surround fashion. Walther and Koch's [33] well-known saliency approach works on 2D images rather than 3D models. Fig. 8 shows the generated saliency maps and gaze points for a frame of shown animations. Since the 3D methods work directly on the models and do not assign any saliency value to the background (contrary to the 2D method), we omitted the backgrounds in our analysis for a fairer comparison.

Fig. 9 shows the results of our user study. For all cases, our model well predicts salient regions and has a high KL divergence value. In one of the cases, the KL divergence of Bulbul et al.'s animated saliency model performs similar to our proposed model. Predictably, for the animated meshes used in the experiment, the saliency computation models considering motion properties are better at saliency estimation than the static case. The proposed model is also superior to Bulbul et al.'s model in general. Another interesting case is the comparison of the 2D and 3D saliency approaches. The image-based saliency approach performs close to random for the Hand and Toasters scene. However, for the more realistic Fairy scene, it performs similar to the static mesh saliency approach regarding KL divergence.

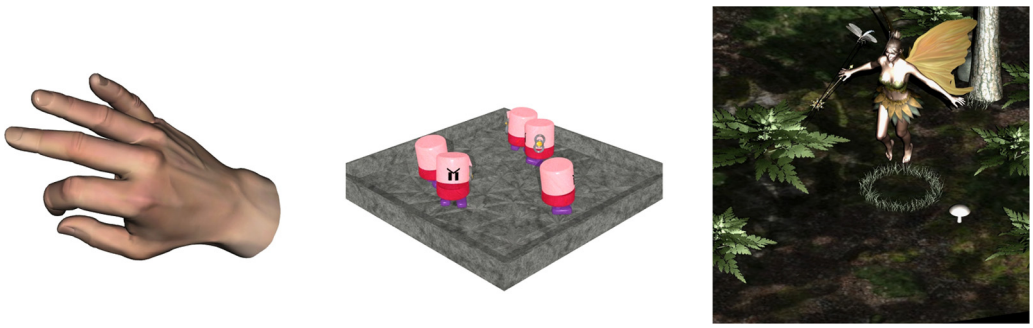

Fig. 7. Three animated meshes from The Utah 3D Mesh Repository are used in the experiment. Left to right: Hand, Toasters, and Fairy.

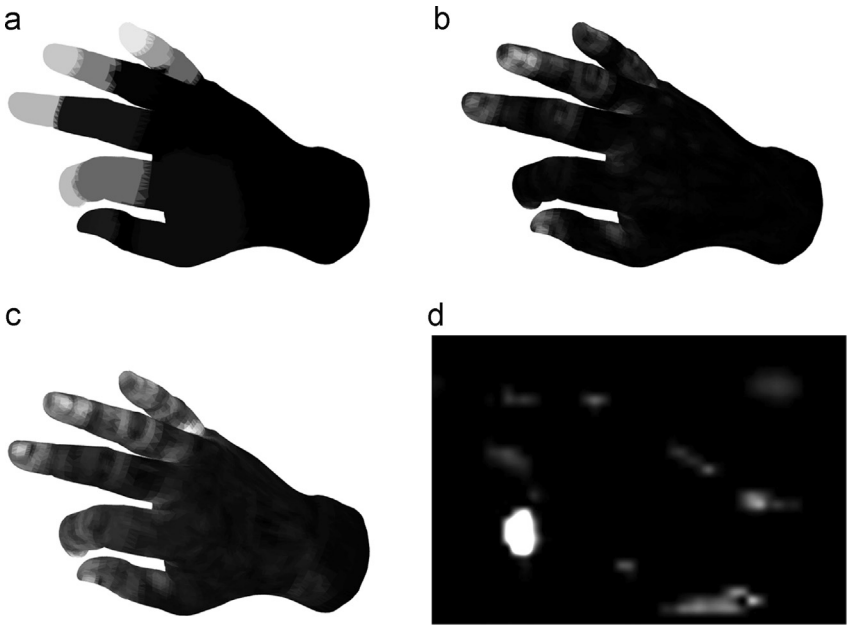

Fig. 8. Saliency maps for a frame of hand animation: (a) proposed model, (b) Lee et al.'s model [14], (c) Bulbul et al.'s model [23], and (d) image-based model [33].

5. Discussion

Despite the undeniable dominancy of task-based top-down attention, object features also affect human gaze directions in a bottom-up fashion. Lee et al.'s mesh saliency approach [14] is one of the keystone studies regarding the saliency calculation of 3D mesh models. The current study proposes a spatial saliency estimation according to the curvature properties of mesh models and finds that in addition to the significance of 3D models' spatial features, their temporal features are also very significant in attracting attention.

In [23], Bulbul et al. estimate motion-based saliency of 3D animated meshes similar to Lee et al. [14]. In that study, the difference of mean velocity between a small and a large neighborhood is used to determine the motion-based saliency. Since all vertices forming part of the 3D mesh move together and have the same motion attributes, such an approach promotes joints, i.e., where the mesh model bends, and down-weighs the saliencies of the centers of the parts. Thus, we propose a solution that regards all vertices belonging to a perceptually meaningful part of a 3D mesh (e.g., a hand) as a single unit. We form the perceptually meaningful parts with a Gestalt approach, considering that the HVS perceives objects as patterns organized by similar attributes, such as motion. This clustering approach works better for animated models composed of rigid parts and for natural objects with a skeletal structure than for abstract shapes such as clothes; the latter do not have parts with perceptually distinguishable motion behaviors.

After the clustering phase, we use a saliency estimation method based on objects' motion states to determine the visual attractiveness of each cluster. Overall, the results confirm that in terms of motion, the proposed solution detects salient regions better than the compared models do. If the same saliency model were applied to each vertex of an animated mesh without clustering, the calculated saliencies would not be much different, considering that vertices in the same cluster share a similar motion behavior. However, clustering the vertices provides a huge performance advantage for the proposed method; considering the notably large numbers of vertices in a 3D mesh model, calculating the neighborhoods of each vertex to find its saliency takes a long time. In the proposed model, the number of units to work on decreases from thousands (the number of vertices) to tens (the number of clusters) after the motion-based clustering phase. This also makes it possible to compute motion saliency in real time. A viewpoint-dependent real-time saliency computation framework could be a promising future work direction.

How to combine different channels of saliencies (e.g., motion-based, appearance-based, or geometry-based) into a single saliency measure is an important problem. Itti and Koch's normalization operator [21], which promotes unique features and suppresses more-frequent features, is a powerful method for combining different saliency scales and channels, and is employed by Bulbul et al. [23]. However, this method requires finding local maxima that are applicable to 2D images and vertex-based saliency calculations performed on manifold mesh models, but it is not directly applicable to multiple models or vertex clusters. A detailed analysis of the effect of various feature channels on saliency and how to combine these channels for 3D mesh models could form another future work area.

A possible approach to estimating saliency of 3D models could be to use 2D saliency metrics directly on the rendered objects. In our comparison of 3D saliency estimation methods to Walther and Koch's well-known 2D saliency method [33], we found that for a single model (which does not cover the whole image) background becomes a problem for image-based saliency calculation. However, for a synthetic and fairly realistic-looking scene with textured objects, our user study shows that an image-based saliency metric could work. Working directly on the final scene after all illumination calculations have been completed is an advantage of image-based methods; however, these methods are not generalizable to all viewpoints and differences in scene structure. Two-dimensional and 3D saliency calculation methods could be compared in more detail in a future study, and it may be possible to come up with a hybrid solution that combines the advantages of both.

6. Conclusion

In this study, we proposed a motion-based saliency approach for 3D animated mesh models. This model considers a 3D model as

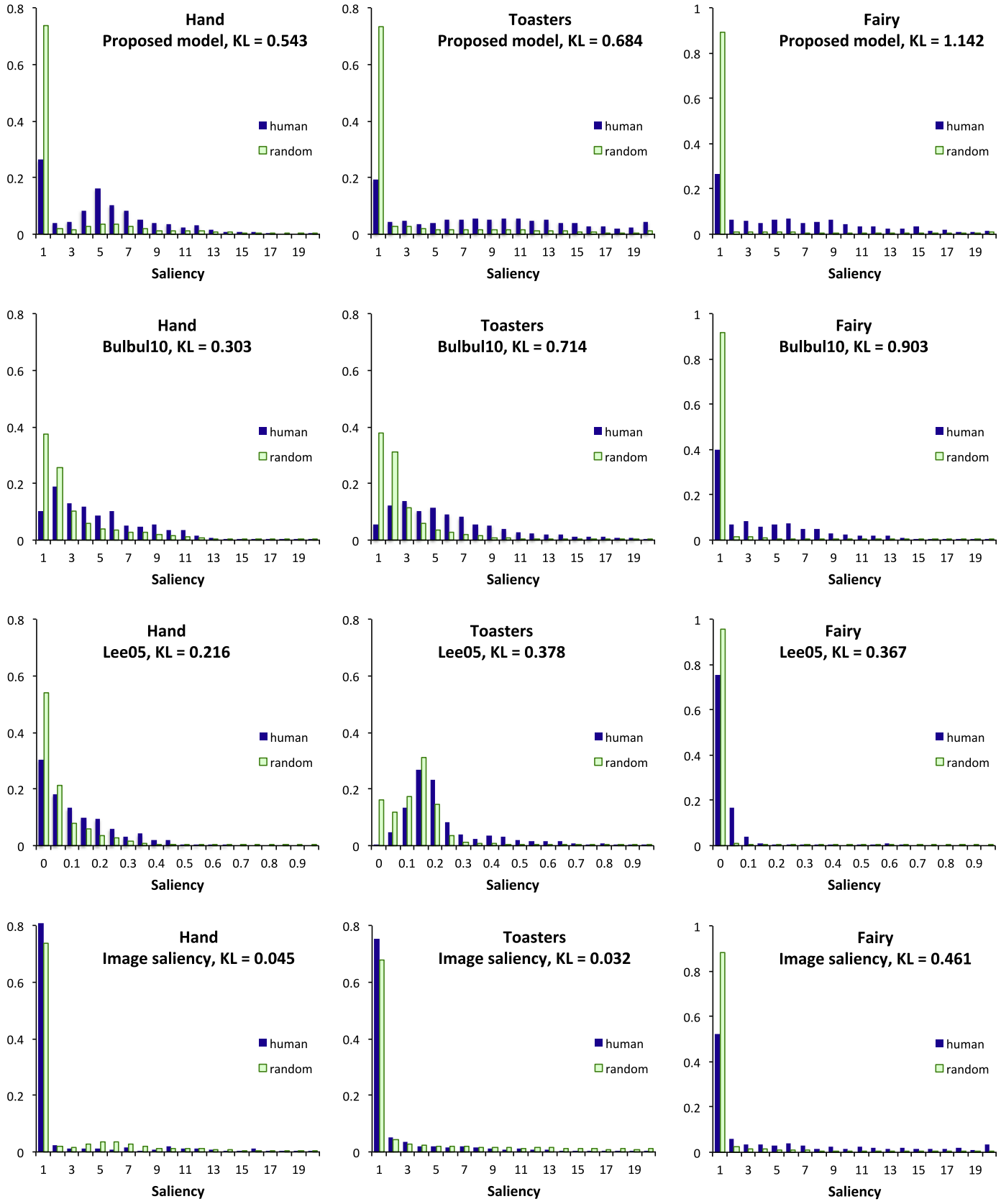

Fig. 9. KL Divergence analysis: saliency distributions of human fixation points vs. saliency distributions of random fixation points.

a combination of separate objects and finds the saliency of each object. To partition the vertices of a 3D mesh model, we also proposed a motion-based clustering method that forms perceptually meaningful groups of vertices.

We performed a formal experiment to evaluate the proposed approach. We compared our model to several existing methods and, in general, our model had higher KL divergence values than the compared models, which verifies its successful results.

Acknowledgments

The horse and camel gallop animations were made available by Robert Sumner and Jovan Popović from the Computer Graphics Group at MIT. The hand, toaster, fairy, and running human (Ben) models were obtained from The University of Utah 3D Animation Repository. Dance model is available at Khaled Mamou's website (http://khaledmammou.com/). We are grateful to Rana Nelson for proofreading and suggestions. Also, we would like to thank all subjects who participated in the experiments. This work is supported by the Scientific and Technical Research Council of Turkey (TUBITAK, Project number: 110E029) and performed when the authors were at Computer Engineering Department of Bilkent University.

Appendix A. Supplementary material

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2014.04.003.

References

[1] Itti L, Koch C. Computational modeling of visual attention. Nat Rev Neurosci 2001:194–203.
[2] Yarbus AL. Eye movements during perception of complex objects. New York, NY: Plenum Press; 1967. p. 171–211.
[3] Navalpakkam V, Itti L. Search goal tunes visual features optimally. Neuron 2007;53(4):605–17.
[4] Simons DJ, Chabris CF. Gorillas in our midst: sustained inattentional blindness for dynamic events. Perception 1999;28(9).
[5] Posner MI, Cohen Y. Components of visual orienting. Hillsdale, NJ: Erlbaum; 1984. p. 531–56.
[6] Frisby JP, Stone JV. Seeing motion, part I. 2nd ed. Cambridge, MA: The MIT Press; 2010. p. 324–53 [chapter 14].
[7] Reichardt W. Autocorrelation, a principle for the evaluation of sensory information by the central nervous system. In: Rosenblith WA, editor. Principles of sensory communications. New York: John Wiley; 1961. p. 303–17.
[8] Andersen RA. Neural mechanisms of visual motion perception in primates. Neuron 1997;18:865–72.
[9] Koffka K. Principles of gestalt psychology. London: Routledge; 1935.
[10] Abrams RA, Christ SE. Motion onset captures attention. Psychol Sci 2003;14:427–32.
[11] Yantis S, Hillstrom AP. Stimulus-driven attentional capture: evidence from equiluminant visual objects. J Exp Psychol Hum Percept Perform 1994;20(1):95–107.
[12] Hillstrom AP, Yantis S. Visual motion and attentional capture. Percept Psychophys 1994;55(4):399–411.
[13] Itti L, Koch C, Niebur E. A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Pattern Anal Mach Intell 1998;20(11):1254–9.
[14] Lee CH, Varshney A, Jacobs DW. Mesh saliency. ACM Trans Graph (Proc SIGGRAPH'05) 2005;24(3):659–66.
[15] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of ACM SIGGRAPH '97; 1997. p. 209–16.
[16] Liu YS, Liu M, Kihara D, Ramani K. Salient critical points for meshes. In: Proceedings of ACM symposium on solid and physical modeling; 2007. p. 277–82.
[17] Kim Y, Varshney A, Jacobs DW, Guimbretière F. Mesh saliency and human eye fixations. ACM Trans Appl Percept 2010;7(2):12:1–13.
[18] Leifman G, Shtrom E, Tal A. Surface regions of interest for viewpoint selection. In: CVPR 2012.
[19] Feixas M, Sbert M, González F. A unified information-theoretic framework for viewpoint selection and mesh saliency. ACM Trans Appl Percept 2009;6(1):1:1–23.
[20] Longhurst P, Debattista K, Chalmers A. A GPU based saliency map for high-fidelity selective rendering. In: Proceedings of the fourth international conference on computer graphics, virtual reality, visualisation and interaction in Africa; 2006. p. 21–9.
[21] Itti L, Koch C. A saliency-based search mechanism for overt and covert shifts of visual attention. Vis Res 2000;40(10–12):1489–506.
[22] Chen X, Saparov A, Pang B, Funkhouser T. Schelling points on 3d surface meshes. ACM Trans Graph 2012;31(4):29:1–12.
[23] Bulbul A, Koca C, Capin T, Güdükbay U. Saliency for animated meshes with material properties. In: Proceedings of the seventh symposium on applied perception in graphics and visualization. APGV '10; New York, NY, USA: ACM; 2010. p. 81–8.
[24] Arpa S, Bulbul A, Capin T. A decision theoretic approach to motion saliency in computer animations. In: Allbeck J, Faloutsos P, editors. Motion in games. Lecture notes in computer science, vol. 7060. Berlin, Heidelberg: Springer; 2011. p. 168-79. ISBN 978-3-642-25089-7.
[25] Wald I. The Utah 3d animation repository. URL: ⟨http://www.sci.utah.edu/~wald/animrep/⟩.
[26] Nishida S. Advancement of motion psychophysics: review 2001–2010. J Vis 2011;11(5).
[27] Yee H, Pattanaik S, Greenberg DP. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. ACM Trans Graph 2001;20:39–65.
[28] Howlett S, Hamill J, O'Sullivan C. Predicting and evaluating saliency for simplified polygonal models. ACM Trans Appl Percept 2005;2(3):286–308.
[29] McDonnell R, Larkin M, Hernández B, Rudomin I, O'Sullivan C. Eye-catching crowds: saliency based selective variation. In: ACM Transactions on Graphics; vol. 28. New York, NY: ACM; 2009. p. 55.
[30] Hou X, Zhang L, Dynamic visual attention: searching for coding length increments. In: NIPS'08; 2008. p. 681–88.
[31] Itti L. Quantifying the contribution of low-level saliency to human eye movements in dynamic scenes. Vis Cogn 2005;12(6):1093–123.
[32] Peters RJ, Itti L. Applying computational tools to predict gaze direction in interactive visual environments. ACM Trans Appl Percept 2008;5(2) (Article 8).
[33] Walther D, Koch C. Modeling attention to salient proto-objects. Neural Netw 2006;19:1395–407.