Vis Comput (2017) 33:1357–1369
DOI 10.1007/s00371-016-1231-2

ORIGINAL ARTICLE

Kinetic depth images: flexible generation of depth perception

Sujal Bista[1] · **Ícaro Lins Leitão da Cunha**[2] · **Amitabh Varshney**[1]

Published online: 6 May 2016

Abstract In this paper we present a systematic approach to create smoothly varying images from a pair of photographs to facilitate enhanced awareness of the depth structure of a given scene. Since our system does not rely on sophisticated display technologies such as stereoscopy or auto-stereoscopy for depth awareness, it (a) is inexpensive and widely accessible, (b) does not suffer from vergence - accommodation fatigue, and (c) works entirely with monocular depth cues. Our approach enhances the depth awareness by optimizing across a number of features such as depth perception, optical flow, saliency, centrality, and disocclusion artifacts. We report the results of user studies that examine the relationship between depth perception, relative velocity, spatial perspective effects, and the positioning of the pivot point and use them when generating kinetic-depth images. We also present a novel depth re-mapping method guided by perceptual relationships based on the results of our user study. We validate our system by presenting a user study that compares the output quality of our proposed method against other existing alternatives on a wide range of images.

Keywords Kinetic depth effect · Depth perception · Aesthetics in visualization · Visualization for the masses

✉ Sujal Bista
sujal@umiacs.umd.edu

[1] Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, USA

[2] Universidade Federal Rural de Pernambuco - Unidade Academica de Garanhuns, Garanhuns, PE, Brazil

1 Introduction

The kinetic-depth effect (KDE) is the perception of the three-dimensional structure of a scene resulting from a rotating motion. First defined by Wallach and O'Connell [53], the kinetic-depth effect has been used widely. Images that exhibit KDE are commonly found online. These images are variously called Wiggle images, Piku-Piku, Flip images, animated stereo, and GIF 3D. We prefer to use the term *Kinetic Depth Images* (KDI), as they use the KDE to give the perception of depth. In 2012, Gizmodo organized an online competition to reward the KDI submission that best provided a sense of depth [55]. Recently, the New York Public Library published a collection of animated 3D images online [18]. Flickr has a large number of groups that discuss and post animated 3D stereo images. Also, online community-based galleries that focus on KDI can be found at the Start3D website [8]. With the increasing availability of stereo and lightfield cameras (such as the Lytro), the use of the KDE to express depth on ordinary displays is rising rapidly.

Although the basic form of the KDE is trivial to implement for a virtual environment where the 3D geometry and lighting are known, excessive motion and reduced depth perception mar the visual experience unless proper care is taken. In fact, accomplishing a high-quality KDE from a pair of photographs is not trivial and has several interesting nuances, as outlined in this paper, which should be useful for any practitioner wishing to use this technique for facilitating depth awareness. In this paper we describe an algorithm and its accompanying system, for facilitating depth awareness through KDE using only a pair of photographs or images.

The use of the KDE is a viable alternative to the use of stereoscopic and autostereoscopic displays: (i) the KDE provides monocular depth cues that allow us to experience depth, even with one eye, which accommodates people who suffer

from various monocular disorders, (ii) the depth is perceived due to the rotation of the object and does not require any special device (glasses or lenses) to view; this works with any display and can be easily shared online, (iii) unlike stereoscopic and auto-stereoscopic displays, KDE animations do not suffer from vergence-accommodation conflicts, and (iv) depth perception achieved by the KDE can exceed that of the binocular depth perception as the disparity provided by binocular vision is limited compared to the angular rotation that can be produced by KDE [12,35].

Although there are a large number of KDE-based images online, most of them are manually created by the artist, require tedious user input to create them, or suffer from visual artifacts introduced by the automated systems used to create them. Manual creation of these images may seem simple, but it is difficult for average users to make their own high-quality KDE-based images. A lot of KDI suffer from artifacts caused by abrupt changes in motion, color, or intensity, as well as alignment errors and excessive motion.

The following are our contributions to this paper:

1. Given a stereo image or an image/depth pair as an input, we automatically optimize a KDE-based image sequence by taking human depth perception, optical flow, saliency, radial components, and disocclusion artifacts into consideration;
2. We report the results of a user study that examines the relationship between depth perception, relative velocity, spatial perspective effects, and the positioning of the pivot point and use them to generate KDI;
3. We present a novel depth re-mapping method guided by image saliency and the perceptual relationship found in our user study to reduce excessive motion in KDI.

2 Background

The kinetic depth effect (KDE) is defined as the perception of the three-dimensional structural form of an object when viewing it in rotational motion [12,53]. As an observer moves, the nearby objects are seen from different angles. To experience KDE, the optical-flow-driven motion cue has been found to be very important [14,43,46] and can be experienced from just two photographs taken from different views [11,26]. Another effect that gives the perception of depth is the stereo kinetic effect (SKE). Although SKE and KDE both give the perception of depth, they are quite different. SKE gives a perception of depth when viewing a 2D pattern as it is rotated in the view plane (fronto-parallel) [41], whereas in KDE the depth perception arises from rotating the object along an axis. Motion parallax is another monocular depth cue that is experienced through the relative motion of the near and far objects [12]. Generally, KDE is associated with the rotational viewing of objects whereas motion parallax is associated with the translational viewing of objects. When an observer fixates on an object and makes small rotational or translational motions, both KDE and motion parallax are similar [12].

3 Related work

Although a few tools are available online to generate the KDE from image pairs, we found them to be largely ad-hoc techniques with variable quality. We have not come across any previous work in the technical literature that systematically identifies the various competing constraints in building such tools to achieve an aesthetically superior viewing result. We next give an overview of the various online tools we have come across on the Internet.

The New York Public Library has an online tool, *Stereogranimator*, that converts stereograms into animated 3D images employing user input to align images [18]. The tool allows users to manually rotate and translate stereo images to align them and change animation speed to create animated GIF images that switch between images. With carefully acquired stereo images and with proper user input, the output of this tool can produce good results. Otherwise, the output may contain artifacts created by the abrupt changes in motion, color, or intensity, or through alignment errors.

Wiggle images follow the same principle and are created by taking a pair of stereo images (or a sequence of images) and flipping between them. With careful consideration during the acquisition stage and a very precise manual alignment during the animation generation stage, a reasonably good quality effect can be achieved. However, the process is tedious and requires significant skill in photography and image processing. Rather than relying on the number and quality of input images, we use a judicious mix of computer vision and rendering algorithms to create high-quality animations needed to experience the KDE from a pair of images.

Piku-Piku images from Start3D (www.start3d.com) provides an online tool for generating KDI. A user uploads an image pair to the Start3D server, which then generates a sequence of intermediate images and provides a hyperlink to view the resulting images on its server [8]. To the best of our knowledge, the underlying algorithm for generation of Piku-Piku images, their processing, and their viewing has not been published. A careful study of the animation created by Piku-Piku images shows that the intermediate frames between an input pair of stereo images are computed as a translation from one input image to other. This method does not produce a good results if the input set of stereo images does not happen to fall on the path suited for the KDE. Blending artifacts are

often seen and if the stereo images are not carefully acquired (for example, if shear or slant is present in the stereo pair), then the output is often of a poor quality.

Another similar tool is the *Stereo Tracer* created by Triaxes (www.triaxes.com). This uses an image-depth pair or a stereo-image pair to generate intermediate views to create animation for producing the KDE. To secure a good output tedious manual adjustment is required. These adjustments include (a) depth map adjustment, (b) image alignment, and (c) manipulation of parameters that control parallax and the plane of zero parallax.

Recently, Lytro Camera announced a perspective shift feature on their camera application [32]. Based on user input, they allow users to change the camera perspective computed using the captured light-field data. To our knowledge, the underlying algorithm has not been published.

Zheng et al. [59] presented a method that focuses on automatically creating a cinematic effect that exhibits the parallax effect. They reconstruct the scene from a series of images, fill the occluded regions, and render the scene using a camera path that attempts to maximize the parallax effect while taking into account occluded regions. Rather than trying to recreate cinematic effect, we try to maximize depth perception based on KDE while reducing motion-induced artifacts. KDE uses rotational motion that is different from the cinematic effects presented by Zheng et al. [59].

None of the above approaches take into account depth perception, image saliency, identification of the best rotation axis, identification of good pivot points for fixation, or depth re-mapping to generate high-quality KDE that we have used in our approach. Another significant departure in our approach has been the decoupling of the rendering camera from the acquisition camera. This has allowed us significantly greater freedom in using the standard angular camera motion with substantially fewer visual artifacts as well as using previously unexplored camera motions to achieve the KDE.

4 Overview of our approach

The input to our system is a pair of images that can generate an approximate depth map. Our approach also works with an image and a depth-map pair of the scene available from a camera coupled with a depth sensor such as the Microsoft Kinect.

We will refer to the cameras used in taking the initial images as *input cameras* and the cameras used for generating the animation (to simulate the KDE) as *rendering cameras*. Our system is able to generate this animation through a series of steps as shown in Fig. 1. First, from the input stereo image pair we compute the optical flow and a depth map. After that, we compute parameters needed to generate a KDE by taking into account depth, centrality, saliency, and optical flow. Depth perceived by kinetic motion depends on the relative velocity. Using the result of the user study (that allows us to map velocities to perceived depth which is explained in Sect. 6) and image saliency, we re-map the depth and create a depth mesh. This remapping reduces total motion while enhancing the depth perception. Finally, we visualize

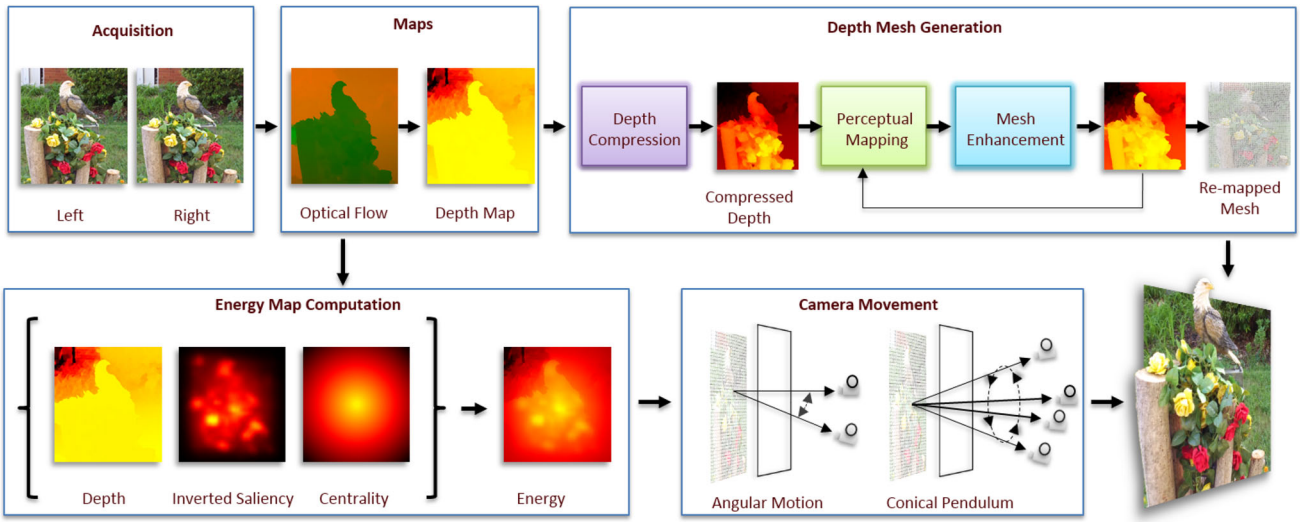


Fig. 1 We first compute the optical flow and the depth map from the input images (generally a stereo pair). We then generate a triangulated depth mesh using the re-mapped depth map which is guided by human perception. We also create an energy map of the image using depth, centrality, and image saliency. By using the depth mesh and the energy map, we generate a high-quality animation to best experience the kinetic-depth effect. Our system has been informed by and validated through user studies

the depth mesh using any desirable rendering camera motions (such as angular or conical-pendulum) to generate the animation needed to experience the KDE. Details of our approach are described in the following sections.

5 Kinetic-depth effect parameters

We generate KDI by calculating proper values for pivot point, rotation axis (shown in Fig. 2), magnitude of angular rotation, and frequency. The pivot point is the look-at point of the rendering cameras as they move about the rotation axis. In this paper, we use two different types of camera motion and each type uses the rotation axis in a slightly different way. We discuss this further in Sect. 9. The pivot point is also a position in the image through which the rotation axis passes. This results in minimal optical flow near the region around the pivot point when the final animation is created.

To create an effective kinetic-depth experience, we need to determine the magnitude of angular rotation and frequency. Wallach and O'Connell [53] used an angle of 42° at the rate of 1.5 cycles/s. Epstein [13] used angles of 15° and higher in his experiment and reported that 15° is sufficient to perceive the KDE. His experiment used the projection of a shadow with no visible shading and texture. Since we are using typical natural stereo images with full texture, we have found that rotation as small as 0.5° about the pivot point is sufficient. This is consistent with human vision, where an average intra-ocular distance is 6.25 cm, which gives about 4° separation at a fixation point 1 meter away. For objects farther away separation is much less. In our results, we perform rotations between 0.5 to 2° around the pivot point. The frequency of rotation used in our system is 2 cycles/s since this has been reported as the optimum temporal frequency by Nakayama and Tyler [36], Caelli [3], and Rogers and Grams [44] .

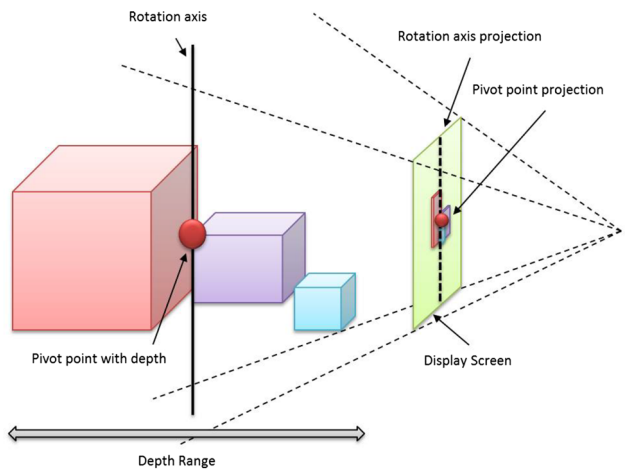


Fig. 2 The pivot point and the rotation axis of the scene are shown in both 3D space and the projection space

Although we keep the rotation at the pivot point small, objects that are close or far might exhibit much higher movement, depending on the scene. According to Ujike et al. [50], 30° to 60°/s on each axis produces the highest motion sickness; roll produces more sickness than pitch and yaw. Taking this into account, we keep the maximum rotation over the entire scene low and keep the change in the vertical axis to a minimum.

The positioning of the pivot point, frequency of rotation, magnitude of angular rotation, and the scene depth range all directly affect the velocity of the stimulus moving on the screen. Gibson et al. [14] and Rogers and Grams [43] have shown that depth perceived by kinetic motion depends on the relative motion. Although some increase in relative motion enhances the perception of depth, excess motion causes motion sickness and reduces the ability to smoothly track objects. Robinson et al. [42] showed that while tracking objects with velocity of 5°/s , the human eye took about 0.22 s to reach the maximum velocity and about 0.48 s to reach the target steady-state velocity. In our experiment, the frequency of rotation is 2 or more cycles and, therefore, to maximize KDE and at the same time reduce unwanted motion-based artifacts, we keep motion below 5°/s. Using depth re-mapping based on results obtained in Sect. 6, we minimize motion artifacts while maximizing the perception of depth.

Several more variables are calculated when we optimize KDI. They are described below.

Relative distance: The relative distance between two vertices v and w is defined by $R_d(v, w) = \|v_d - w_d\|_2$.

Velocity: The velocity of a vertex v is computed by taking camera parameters, rendering motion, and viewing setup into consideration. So, we first compute positions of v in the screen space at times t_0 and t_1. We call them $\grave{v}_{t_0}$ and $\grave{v}_{t_1}$. To do this, we use camera motion parameters to generate a KDE and project them on computer screen. Next, the velocity of v in screen space is computed by equation $\grave{V}_v = \frac{\grave{v}_{t_1} - \grave{v}_{t_0}}{t_1 - t_0}$. Then the velocity $\grave{V}_v$ is converted into angular velocity expressed using view angle by equation $V_v = \grave{V}_v \left(\frac{2 * \tan^{-1}(\text{pixelSize} * .5)}{\text{viewing Distance}} \right)$.

Relative velocity: Relative velocity between two vertices v and w is defined by $R_v(v, w) = V_v - V_w$.

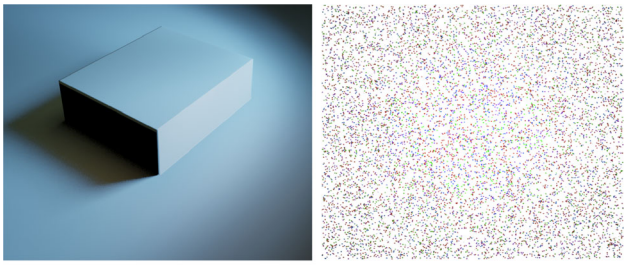
Disocclusion threshold: When viewing a depth mesh with a moving camera, regions that were occluded from the input camera used to create the depth mesh could become visible. As we draw a triangulated depth mesh, these deoccluded regions are filled by the stretched triangles (also known as *rubber sheets*) that span the depth discontinuity from the edge of the foreground object to the background [7,33]. To

minimize the disocclusion artifacts, the rendering camera placement and movement to experience the kinetic-depth effect should be constrained. However, even after performing the optimizations some disocclusion artifacts remain due to the structure of the scene. To quantify amount of perceptible disocclusion when KDI is generated, we calculate disocclusion estimates between two vertices v and w defined by $O_{\text{occ}}(v, w) = \left\| \grave{v}_t - \grave{w}_t \right\|_{\infty} * C_{\text{lab}}(v_{\text{col}}, w_{\text{col}})$ where the maximum screen space difference over the entire KDE animation is multiplied by the function C_{lab} which computes color difference in CIE LAB color space. To determine disocclusion of the entire mesh O_{MeshOcc}, we take k largest O_{occ} and compute average as given by equation $O_{\text{MeshOcc}} = \frac{1}{k} \sum_k O_{\text{occ}}$.

6 Experiment

We performed an experiment to determine the relationship between velocities, positioning, and depth perception. Numerous studies have been carried out to determine what causes motion-based depth perception and its implications on segmentation, depth ordering, and depth magnitude estimation [12,14,26,36,37,43,44,52,56]. Most of them study motion in isolation and focus only on relative velocities. Since we generate a KDE based on stereo images, for us the relationships between depth perception, spatial perspective, and the positioning of the pivot point in the scene is very important. All of these factors influence our depth perception generated by KDE. To our knowledge, there has been no attempt thus far in examining the aggregate relationship among these factors and KDE. Specifically, we examine the combined effect on depth perception from the following factors: (a) the background and foreground of an object placed at the pivot point move in opposite directions, while for an object distant from the pivot point they move in the same direction; (b) the average velocities of the objects placed at the pivot point are much lower than for objects that are distant from the pivot point; (c) the relative velocity between the foreground and the background of an object decreases considerably when receding from both the view point and the pivot point due to the perspective effect, and finally (d) the pivot point has no motion.

We recruited volunteers from our campus and the local community for the experiment. All volunteers had normal or corrected-to-normal vision and were able to perceive KDE. The experiment was performed by following the university IRB guidelines and the participants were compensated nominally for their time and effort. The experiment was conducted in a brightly lit lab using a Dell monitor 2407WFPH with pixel pitch 0.270 mm. The distance between the participant's eye and the monitor was around 0.75 m, which is within the OSHA recommended range [51]. We showed stimuli, which



(a) **(b)**

Fig. 3 The image **a** shows the side view of the object being displayed in the experiment rendered with lighting for illustration purposes. The image **b** shows the stimulus shown to the participants. Without KDE motion, the structure is seen as a flat collection of points

subtended around $20° \times 20°$ visual angle, made of randomly generated dots with various depths. When motion was not present, the randomly generated dots were perceived as a flat fronto-parallel plane (Fig. 3b).

The experiment was done to study the relationship between relative velocity, depth perception, and the positioning of the pivot point. In this experiment, we rendered objects with various depths using randomly generated dots. The objects were composed of a plane with a box either going in or coming out as shown in Fig. 3. The variation of depth between the front and the back of the object changes the relative velocities between them. The objects were placed at various distances from the pivot point, which changes rotational velocity. The objects close to the pivot point experience less velocity. The positioning of the pivot point, scene depth, and spatial perspective affects the relative velocity of pixels on the projected screen. For the study, the relative velocity between the foreground and the background was in the range 0.02°–2.5° visual angle/s. In this study, 11 volunteers participated.

Method: Each participant performed 40 trials with 3 s intervals between trials. There was no restriction on time. Participants were asked to estimate the size of the stimulus using a slider. The specific instruction to them was *Use the slider to show us how much depth you perceive.* The users were then supposed to move the slider from its origin to a distance that visually corresponded to the depth that they were perceiving.

Result: We found that the estimation of the perceived depth between subjects varied dramatically. This is consistent with the previous study [12], where the researchers reported a high variation in perceived depth. However, when depth values are normalized per subject, a distinct pattern emerged. We computed the average of the normalized depth for all subjects for each distance from the pivot point, as shown in Fig. 4. The participants perceived increased depth between relative velocities of 0.2°–1.25° visual angle/s; for higher relative velocities, the perceived depth remained constant.

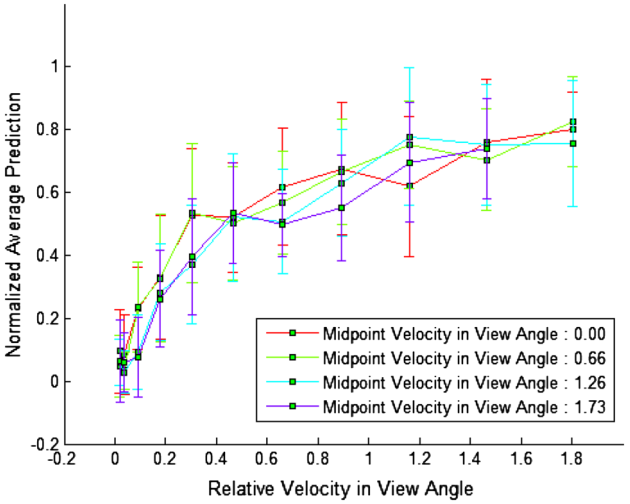


Fig. 4 The figure shows the mean and standard deviation of the normalized depth perceived by subjects. When the objects are placed at various distances from the pivot point, the perceived velocity of the object changes. We show separate curves for each distance we tested and list the midpoint velocity. Between relative velocities of 0.2° to 1.25° visual angle/s, participants perceived increased depth. For higher relative velocities, the perceived depth remained constant

Thus, we consider 1.25 as the maximum desirable relative velocity. This is taken into consideration while generating the depth mesh. We performed a two-way analysis of variance (ANOVA) between the distance from the pivot point and the relative velocity in the field of view (view angle). The results of ANOVA for the distance from the pivot point are $[(F(3, 400) = 3.16, P = 0.0248]$, the relative velocity are $[(F(9, 400) = 108.94, P < 0.0001]$, and the interactions between them are $[(F(27, 400) = 0.96, P = 0.5208]$. These values indicate that both the distance from the pivot point and the relative velocity affect the perceived depth, but there is no evidence of an interaction between the two. Please look at the accompanying video for a visual explanation of the results.

Perceptual maps: Our experiment and previous studies have shown that relative velocity is an important motion cue to experience KDE [14,43,46]. By applying Gaussian filter to smooth the results from this experiment, we compute a set of curves that are used to map depth perception to relative velocity and vice versa. We generated two sets of curves to approximate the results from the experiment as shown in Fig. 4. The first set of curves maps the relative velocities to normalized depth perception (RV2NDP map) and the second set maps the users' normalized depth perception to the relative velocities(NDP2RV map). The inverse of the RV2NDP map is the NDP2RV map. The curves within each set show how the relation between relative velocity and depth perception changes as the midpoint velocity is changed. This mapping gives an empirical relationship between the normalized depth perception and the relative velocity.

7 Energy map computation

As the pivot point experiences minimal optical flow, it is preferable to locate the pivot point at a salient region of the scene. If the pivot point salient region has text or a face, it becomes easier to read or identify it when the region is not exhibiting high optical flow due to the rendering camera movement. It is also preferable to have the pivot point in the middle of the scene to minimize the optical flow at the scene boundaries. If the pivot point is chosen at a scene boundary (as an extreme example), the opposite end of the scene will exhibit excessive motion that can create visual discomfort and can also give rise to significant disocclusion artifacts.

To quantify and incorporate such considerations, we determine the placement of the pivot point by computing an energy map. We seek a global minimum on the energy map to determine the kinetic-depth parameters:

$$E(x, y) = E_{\mathrm{d}}(x, y) + E_{\mathrm{s}}(x, y) + E_{\mathrm{r}}(x, y), \tag{1}$$

where (x, y) refers to the position of the pivot point in the original image (projection space) and $E_{\mathrm{r}}(x, y)$, $E_{\mathrm{d}}(x, y)$ and $E_{\mathrm{s}}(x, y)$ are the radial, depth, and saliency energy functions, respectively. Each energy function component is further discussed below.

Depth energy: We use the depth energy to express a preference for positioning the pivot point on regions that are close to the middle of the depth map. This minimizes the visibility of the occluded regions during the rendering camera movements and also lowers the amount of motion when the final animation is generated. We obtain the depth map of a given image set by calculating its optical flow using Sun et al.'s [48] algorithm. Details about depth map calculation are described in Sect. 8. After calculating the depth map we can calculate the depth energy as $E_{\mathrm{d}}(x, y) = \|P_{\mathrm{d}}(x, y) - D_{\mathrm{m}}\|_2$, where $P_{\mathrm{d}}(x, y)$ refers to the depth value of the pixel at (x, y) and D_{m} is the median depth of the scene. The second image of Fig. 5 shows the depth map used in $P_{\mathrm{d}}(x, y)$.

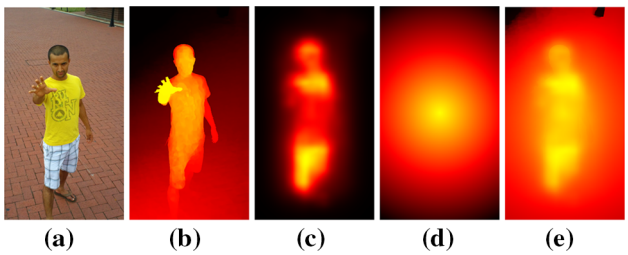


Fig. 5 Energy components. Here we show **a** the original image, **b** the depth map, **c** the saliency component, **d** the radial component, and **e** the final computed energy map. In these maps, *yellow* regions are associated with lower energy. We choose the pixel with the lowest energy to be the pivot point

Saliency energy: As mentioned earlier, we desire to position the pivot point on a salient region of the scene. These salient regions are different in color, orientation, and intensity from their neighbors and can be found using the image saliency algorithm proposed by Itti et al. [21]. The saliency map represents high-saliency regions with high values. When calculating the saliency energy of the pivot point, we invert the saliency values so that the most salient regions are represented with the lowest values. Our equation for calculating the saliency energy is $E_s(x, y) = [1 - P_s(x, y)]$, where P_s refers to the saliency value of the pixel. The third image in Fig. 5 illustrates the saliency map used in $P_s(x, y)$.

Radial energy: To express a preference for a centralized pivot point, we use a radial energy function as one of the energy components. Essentially, the closer the point is to the center, the less radial energy is associated with it. The radial energy is calculated as $E_r(x, y) = P_r(x, y)$, where P_r refers to the radial value of the pixel defined by the Euclidean distance between the point and the image center. The fourth image in Fig. 5 shows an example of the radial map.

The radial component $P_r(x, y)$ depends upon the dimensions of the image, but the saliency component $P_s(x, y)$ and the depth component $P_d(x, y)$ depend upon the scene. To take these factors into account, we can add weights while calculating the energy function. Assigning a higher saliency weight will increase the importance of the salient regions, whereas higher depth weight and radial weight will give greater priority to the image center and lower the total optical flow between frames. Figure 5 shows an example of energy map calculation. We compute the energy of all the pixels and find the position (x, y) that has the lowest energy. Then the pivot point is defined by the coordinate $[(x, y, P_d(x, y)]$.

8 Mesh generation

We generate the depth mesh by first approximating the scene depth, followed by an optimized compression for KDE. Then, we perform re-mapping of the depth mesh by taking perception into account. Finally, we enhance the depth mesh so that the motion is constrained to a desired range and disocclusion artifacts are minimized.

8.1 Scene depth approximation

Generally the number of input images or photographs is smaller than the desired number of views. Also, the parameters used by the rendering cameras are often different from the input cameras. Due to these reasons, a number of additional intermediate frames need to be generated. There are numerous ways of generating intermediate frames, such as basic interpolation between input images, flow-field interpolation [61], image-based rendering [2,6], and structure approximation [1,5,15,17,31,34,38,45,58,60] to create depth mesh.

In our approach, we use a depth-image-based rendering to generate high-quality intermediate frames needed for achieving the KDE. This is computationally efficient and allows sufficient flexibility in the choice of the rendering camera parameters. For every pair of images, we first calculate the depth map. Although there are numerous methods to compute the depth map, we decided to approximate depth based on the optical flow between the input image pair by using the inverse relation defined as $d = f\frac{t}{o}$, where d is the depth, f is the focal length, t is the distance between the camera positions, and o is the optical flow magnitude of the pixel. Since we do not know the camera parameters f and t, we recover depth up to a projective transformation. Optical flow between images is a vector field. Objects at the zero plane will have zero optical flow. Objects that are in front and behind the zero plane will have optical flow vectors facing in the opposite directions. Taking either the maximum or minimum optical flow and adding it to the entire vector field will shift the zero plane to either the closest or the furthest depth. We then convert optical flow map to depth map.

8.2 Optimized depth compression

Depth range on a raw depth map is usually very high and contains a lot of artifacts. Directly using a raw depth map will cause excessive motion that is visually disconcerting. One naïve solution to this problem is to perform simple scaling of the depth map to fit a certain range. However, this will compress both important and unimportant regions of the scene uniformly. Extensive amount of research has been conducted in visual saliency and perceptual enhancements [9,10,16,20,21,23,24,28,29,57]; however, use of motion makes our situation unique. We want the salient regions of the scene to occupy a larger share of the depth range while compressing non-salient regions and artifacts. An analogous problem arises in stereo viewing and relief mapping, where disparity or depth compression is needed [22,27,30,40,54]. In KDI, compression requires global consistency of the scene depth as the depth mesh is viewed from different angles; otherwise, the depth inconsistencies will be perceived easily. The disparity histogram and saliency map have been used by Lang et al. [27] to compress disparity for stereo viewing. In the slightly different problem of video retargeting, manual constraints have been used to enforce feature preservation while compressing the frames [25].

We perform depth compression based on the image saliency and use cues from carefully chosen edges to enforce feature preservation. Our approach for compressing depth is

similar to that of Lang et al. [27]. We would like to account for saliency in compressing depth. To compress depth, we first divide the depth range of the mesh evenly into k intervals $\{r_0, r_1, ...r_{k-1}\}$. We would like to have a greater compression for depth intervals that are largely empty or have low-salience vertices. To achieve this, we build a histogram over the depth intervals, r_x in which we use vertex counts weighted by their respective saliencies (each vertex's saliency is in the range 0 to 1). Next, we compute a compressed value s_x for the interval r_x as follows:

$$s_x = \min\left(\frac{h_x}{g * \max(h_0, h_1, ...h_k - 1)}, 1.0\right), \tag{2}$$

where s_x is the size of the interval x after using saliency compression, and g is a constant which gives extra rigidity to salient depth intervals. In practice, we found that $g = 0.4$ gives a proper balance between compression and rigidity of depth intervals. This non-linear mapping compresses the intervals that are less salient.

Only using saliency to compress the depth map will cause features, such as lines and curves, to change. This is because depth intervals are compressed non-linearly. Some constraints have to be placed to preserve features. We use information of carefully chosen edges to enforce loose feature preservation. We first calculate edges in the image using Canny edge detection [4]. If needed, more sophisticated edge detectors can be easily integrated. We filter edges that are very short in length to remove noise. After that, we perform additional filtering to remove edges that lie at the border region of objects at various depths. This is done by removing edges with very high gradient on their depth values. This step is important because the depth map contains errors/artifacts and the depth approximation close to the depth boundaries is less reliable. Edges that are left after filtering will be longer and will have more reliable depth values. These edges are used for feature preservation. We find the depth interval associated with each of these filtered edges and uniformly distribute depth among them as $s_x^* = \frac{\sum_{n=i}^{j} s_n}{j-i}$, where s_x^* is the new size of the depth interval x after using saliency compression with feature preservation; i and j are the depth interval associated with the two endpoints of a filtered edge and $x \in [i, j]$. Finally, using the compressed depth, we create compressed depth mesh M_{Comp}.

8.3 Perceptual re-mapping

Mesh re-mapping: We generate the re-mapped depth mesh by taking depth perception into account. Since we would like the viewers to perceive depth that is close to the 3D structure we present, we use the relative depth (separation) computed from the compressed depth mesh M_{Comp} as an input for the NDP2RV map (from Sect. 6) to generate a relative velocity map needed to perceive the desired depth. This step is done for each vertex v in M_{Comp}. For each v, let V_{n} be the set of its four connected neighbors such that $V_{\text{n}} \subseteq M_{\text{Comp}}$. We created four vertex-neighbor pairs (v, w) where w is in the set V_{n}. Then for each pair (v, w), we compute the relative depth $R_{\text{d}}(v, w)$ between them. The relative depth is scaled so that it is within the maximum depth range allowed in the scene. At first, maximum depth range is initialized to the depth range that results in a relative velocity that is equal to the maximum desirable relative velocity which is explained in Sect. 6. The scaled depth is then used as an input for the NDP2RV map to get the relative velocity $V_{\text{p}}(v, w)$ between the pair. We call $V_{\text{p}}(v, w)$ a perceptual relative velocity because it is based on the perceptual relationships explained earlier. The $V_{\text{p}}(v, w)$ between a vertex pair is necessary to perceive the relative depth $R_{\text{d}}(v, w)$ between them. In other words, to perceive a depth that is equal to $R_{\text{d}}(v, w)$ between a vertex pair, we need the relative velocity between them to be equal to $V_{\text{p}}(v, w)$.

Using the perceptual relative velocities between vertices, we can compute their separation (we call this perceptual separation $S_{\text{p}}(v, w)$). We move one of the vertices in (v, w) along the line of the view point and find separation $S_{\text{p}}(v, w)$ that results in the desired $V_{\text{p}}(v, w)$. The relative velocity between a pair of vertices is computed by projecting each vertex on the display screen using a standard projection matrix, then calculating the instantaneous velocity (described in Sect. 5) for each vertex, and finally finding the difference between the velocities. This process is accelerated by a binary search algorithm. Since this step is performed by only using local data on the distances between neighbors (v, w), an extra step to make a globally consistent mesh is required. This is done by minimizing a 1D problem. We first discretize depth of the scene into 255 bins. Then for each (v, w), we find associated discretized depths d_v and d_w based on their depth in M_{Comp}. We then add a link between d_v and d_w that specifies the $S_{\text{p}}(v, w)$. This process can be pictured as a spring mesh where the links added will act like a spring and the discretized depths are the locations where the springs are attached. Since the $S_{\text{p}}(v, w)$ is locally consistent, some links will try to contract the distance between the discretized depths while some will try to expand them. We define $* \; d(v, w) = \mid d_v - d_w \mid$. For all (v, w), we find $d(v, w)$ that minimizes $\sum \left\| d(v, w) - S_p(v, w) \right\|_2$ to perceptually re-map the depth mesh M_{P} to make it more globally consistent.

8.4 Depth mesh enhancement

In Sects. 8.2 and 8.3, we have carried out local depth enhancement using local per-pixel neighborhoods. However, this can cause the range of the motion for the entire image to be too low or high. In this section we carry out a global optimization

for a more pleasant viewing experience. Depending on the rendering parameters and the depth range of the scene, depth mesh can be compacted or expanded to allow enhanced depth perception. Also, depth has to be adjusted to reduce disocclusion artifacts. In Sect. 6 we mention that when using KDE, we have a limited perception of depth, which is highly dependent on the relative velocity. The relative velocity of objects depends on the relative distance between them, the placement of the pivot point, and the camera movement. However, there is a tradeoff between the amount of motion in the scene due to relative velocity and perception of depth. The results of Sect. 6 give us the maximum desirable relative velocity, taking into account the tradeoff. Using the maximum desirable relative velocity, perceptual depth mesh, and placement of the pivot, we find the maximum depth range $D_{\max}$ allowed in the scene. In other words, given perceptual depth mesh and placement of the pivot, we find depth range $D_{\max}$ which results in a relative velocity equivalent to the maximum desirable relative velocity between the front and back extremes of the mesh. This is done by searching for the depth range that results in the desirable relative velocity. This process is accelerated by a binary search algorithm. Then we define the minimum depth range allowed in the scene $D_{\min} = D_{\max} * 0.5$.

Once we find the depth range allowed, we focus our attention on finding the depth range that minimizes disocclusion artifacts. In between vertices, we estimate disocclusion by computing relative separation in pixel units between neighboring vertices of the depth mesh when the camera movement needed for KDI is performed. However, when colors between the vertices are the same, disocclusion is not visible and we can ignore disocclusion between these vertices. To approximate the disocclusion of an entire scene O_{MeshOcc}, we take the mean of k-highest disocclusion estimates between vertices whose color is perceptually different as described in Sect 5. When the depth range is small, disocclusion artifacts as well as the perception of depth is reduced. So, to find a proper balance, we compute the depth range by

$$D_{\text{Final}} = \begin{cases} D_{\text{Min}}, & \text{if } (D_{\text{Opt}} \leq D_{\text{Min}}) \\ D_{\text{Max}}, & \text{if } (D_{\text{Opt}} \geq D_{\text{Max}}) \\ D_{\text{Opt}}, & \text{otherwise,} \end{cases} \tag{3}$$

where D_{Opt} is the depth range that has k-highest disocclusion estimates that are less than a user-specified threshold (here we use threshold value of 2). If the difference between D_{Final} and the depth range of M_{P} is small, we simply scale the depth range of M_{P} to match D_{Final}. However, if the depth range is large, we modify the maximum depth range allowed in the scene and start the perceptual re-mapping process again. In Fig. 6, we show a comparison between the raw depth and the depth used to generate final depth mesh after applying all the optimizations stated here.

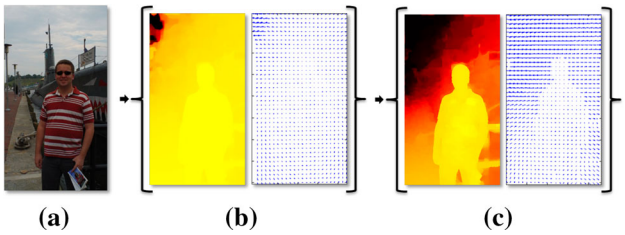


Fig. 6 **a** The original image, the image set **b** is associated with the raw depth, and the image set **c** shows the depth used to generate final depth mesh after depth compression and perceptual enhancements. The depths in **b** and **c** are normalized. In each set we are showing the depth values and the optical flow after the camera motion to depict the value of depth remapping

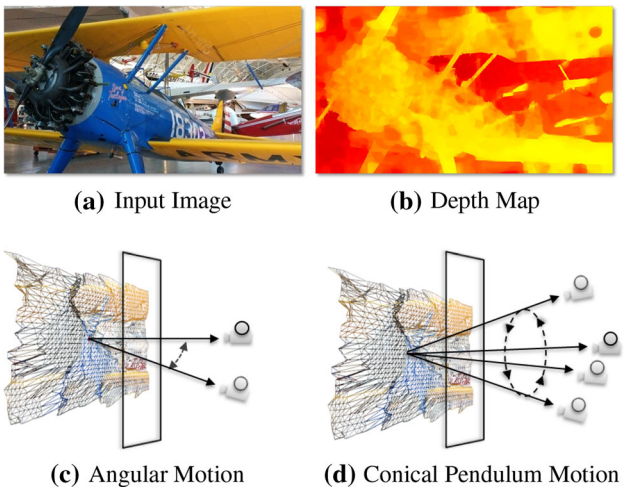


Fig. 7 An illustration using input image (**a**) and optimized depth map (**b**) of the different types of camera movements : the angular motion (**c**) where the camera swivels on a plane perpendicular to the rotation axis while looking at the pivot point, and the conical pendulum motion (**d**) where the camera rotates along a conical surface while looking at the pivot point. Rotation axis is always vertical for the angular motion and is perpendicular to the view-plane for the conical pendulum motion

9 Rendering camera motion

We make use of the pivot point and the rotation axis calculated earlier to compute the rendering camera motion in two ways: angular motion and conical pendulum motion.

Angular motion to experience the KDE involves rotating the camera on a plane perpendicular to the rotation axis while looking at the pivot point as illustrated in Fig. 7c. The angular motion is close to most of the wiggle stereo images found online. However, rather than just flipping between two images and only relying on the sequence of images that were captured by the camera, we generate the intermediate views by rotating the rendering camera positions along an arc subtending a fixed angle at the salient pivot position. Figure 8 shows our results using the angular motion.

Fig. 8 Representation of the angular camera motion

Conical pendulum motion involves rotating the camera along a circle on a plane parallel to the view-plane, as the vector from the camera to the look-at pivot point traces out a cone. Figure 7d illustrates the conical pendulum motion performed by the camera.

10 Rendering and interaction

The depth mesh, along with the camera motion described earlier, is used for rendering the scene. The depth mesh is kept static while the rendering camera moves to generate the KDE. Since we compute the depth mesh, our method allows us to add virtual objects or different layers into the scene at user-specified depth locations. We can, therefore, use the same camera parameters to render additional geometry at the desired locations while also rendering the depth mesh. This makes the kinetic-depth motion consistent for both the depth mesh as well as the additionally-added geometry. Currently, we do not attempt to make lighting and shading seamless and consistent between the virtual object and the scene, but it would be an interesting exercise to attempt to achieve it by estimating the lighting parameters or by using methods such as Poisson image editing [39].

Although we compute the pivot point and the rotation axis automatically as a default, our system also allows the user to change these features as desired. This allows the user to customize the output of the kinetic-depth movement according to their needs. As mentioned earlier, the area around the pivot point has less motion. If there is more than one region that has low energy in a scene, being able to move the pivot point to various locations is critical.

11 Results and discussion

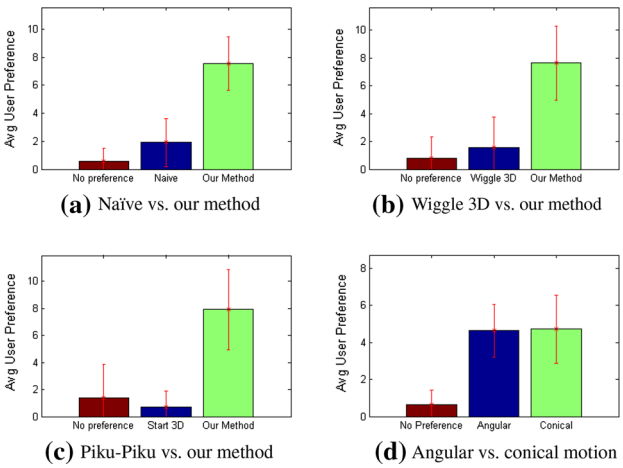
We have implemented our system in C++ and Matlab. For all of our experiments, we use a Windows 7 64-bit machine with an Intel Core i5 2.67 GHz processor, an NVIDIA GeForce 470 GTX GPU, and 8 GB of RAM. We calculate image saliency using the graph-based visual saliency algorithm of Harel et al. [16] implemented in MatLab and use Sun em et al.'s [48] code for optical flow calculations. The rendering of KDI is done at interactive frame rates; however, the creation takes time. The optical flow and saliency approximation algorithms takes majority of the computation time during creation. On average, one megapixel image takes about 4 min. However these pre-processing steps are not currently optimized for speed.

11.1 Subjective evaluation

In order to evaluate the perceptual quality of the generated KDI images, we conducted a user study with 11 subjects who also participated in the earlier user study. Each subject performed four different tests to compare between: (1) Wiggle 3D and our proposed method, (2) the naïve method and our proposed method, (3) Piku-Piku and our proposed method, and (4) angular and conical camera motion. We presented ten examples for each test selected randomly from a larger collection of examples. Each example showed two images side by side. Subjects were asked which image they preferred by taking into account perceived depth and motion. The specific question to the subjects was *Do you prefer the left image, the right image, or have no preference for either?*. The subjects did not know which images were generated using our method.

Naïve vs. our method: The naïve method uses a scaled depth mesh (the magnitude of raw depth range scaled to the same range as our method) with the pivot point selected at the middle of the scene. The same camera motion in rendering was used for both the naïve method and our method. Out of 110 examples, subjects selected the naïve method 21 times, our method 83 times, and had no preference 6 times. In Fig. 9b, we show the user preference per subject. We performed ANOVA with $[(F(1, 20) = 53.24; P < 0.0001]$ which shows that the difference is significant.

Wiggle 3D vs. our method: Generally, wiggle 3D images are created by an artist by manually aligning the most salient



(a) Naïve vs. our method **(b)** Wiggle 3D vs. our method

(c) Piku-Piku vs. our method **(d)** Angular vs. conical motion

Fig. 9 The results of the subjective evaluation. Here we show the average user preference

region of the images. To generate Wiggle 3D image, most salient region of the images is calculated using image saliency algorithm proposed by Itti et al. [21]. Then the zero plane position is selected to coincide with the most salient part of the scene usually located around the central region of the scene. Out of 110 examples, subjects selected Wiggle 3D images 17 times, our method 84 times, and had no preference 9 times. In Fig. 9a, we show the user preference results per subject. We performed ANOVA to analyze the difference between user selection of Wiggle 3D and our method. We got $[(F(1, 20) = 34.21; P < 0.0001]$, which indicates that the difference is significant.

Piku-Piku vs. our method: Here we compared our method with Piku-Piku images from Start3D. Out of 110 examples, subjects selected Piku-Piku images 8 times, our method 87 times, and had no preference 15 times. In Fig. 9c, we show the selection results according to each type of camera motion per subject. We performed ANOVA with $(F(1, 20) = 55.04; P < 0.0001)$ which indicates that the difference is significant.

Angular vs. conical motion: We have compared images generated by two camera motions to find which method subjects preferred. Out of 110 examples, subjects selected angular motion 51 times, conical motion 52 times, and had no preference 7 times. In Fig. 9d, we show the selection results for each type of camera motion per subject. We performed ANOVA with $[(F(1, 20) = 0.02; P = 0.8987]$, which indicates that the difference is not significant.

Discussion: We took a detailed look at the examples where the subjects chose other alternative methods over our method. In most of these examples, we found that the depth range was low and the salient objects were already in the middle of the scene. In these examples, all of these methods generated comparable output. Also, both our method and Piku-Piku images rely on the computed depth maps. Any error in depth maps will affect the final output.

Wiggle3D and Piku-Piku images are highly dependent on the input pair of stereo images to generate the final rendering. They either switch or generate intermediate frames between the input images. Based on the angle of the input camera, the images may contain scaling or shearing. The perception of depth is reduced when noise, scaling, or shearing appears in the animated images. This can be easily observed when watching the animation. However, these subtle artifacts are difficult to show using the vector plot, so we would like to request that the reviewers look at the accompanying video.

11.2 Limitations of our approach

Although the use of the KDE to help understand the three-dimensional structure of a scene is valuable, it also has some disadvantages. First, adding motion could be visually distracting and has the potential to induce motion-sickness. Although this can be considerably reduced by making the camera motion small and smooth and by depth re-mapping, it cannot be completely eliminated. Second, the depth perceived by the KDE is not the same as the depth perceived by binocular disparity. In fact, Durgin et al. [12] have shown in their experiments that depth judgments based on binocular disparity are more accurate compared to depth judgments based on the KDE or motion parallax. However, binocular disparity perceived using stereoscopic displays has its own disadvantages such as the vergence and accommodation conflicts, and visual fatigue [19,47]. To reduce the visual fatigue in stereo displays, various techniques that compress the depth are also used [19], and they too are likely to reduce the accuracy of the depth judgments. Third, our algorithm relies on computing approximated depth maps from a set of stereo images. When the depth map calculation has a significant error, the quality of our animation could also suffer. Thus, both ways of looking at 3D scenes have their own advantages and disadvantages. However, the use of the KDE is simpler and does not require any special devices.

12 Conclusions and future Work

We have presented a method to automatically create smoothly animated images necessary to experience the KDE. Given a stereo image pair or an image-depth map pair as input, we have presented an approach to automatically generate animation that exhibits the KDE. Our approach allows decoupling of the input cameras from the rendering cameras to give us greater flexibility in defining multiple rendering camera motions. We have used two different ways of performing rendering camera motions (angular and conic pendulum motion) to view the resulting scene that minimize visual artifacts by taking into account depth perception, image saliency, occlusions, depth differences, and user-specified pivot points. We have reported the results of user studies that examine the relationship between depth perception, relative velocity, spatial perspective effect, and positioning of the pivot point when generating KDI. We have presented a novel depth re-mapping method guided by perceptual relations based on the results of our user study. And finally, we have presented a subjective evaluation of our method by comparing it against other existing alternatives on a wide range of images.

At present, we have optimized one variable at a time to do depth enhancement. A single optimization across all the pixels is likely to lead to superior results and would be worth exploring. Another future direction is to explore the usage of KDE on modern 3D games to enhance the perception of the 3D structures. KDI may very well become a popular 3D previewing tool for scientific visualization applications where depth awareness is important (such as stereo microscopy) and for consumer-entertainment applica-

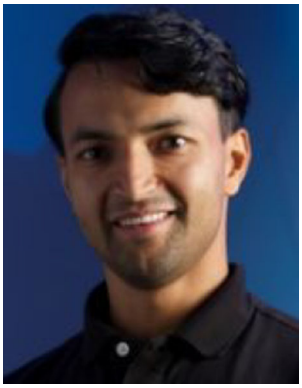
tions including technology-mediated social community sites. Another important group that could benefit from KDI are the people that have lost sight in one eye. Toyoura et al. [49] estimate that number to be around 300 million.

Acknowledgements This work has been supported in part by the NSF Grants 09-59979 and 14-29404, the State of Marylands MPower initiative, and the NVIDIA CUDA Center of Excellence. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors.

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building Rome in a day. In: International Conference on Computer Vision, pp. 72–79 (2009)
2. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 425–432. ACM (2001)
3. Caelli, T.: On the perception of some geometric properties of rotating three dimensional objects. Biol. Cybern. **33**, 29–37 (1979)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell **8**, 679–698 (1986)
5. Chapiro, A., Heinzle, S., Aydın, T.O., Poulakos, S., Zwicker, M., Smolic, A., Gross, M.: Optimizing stereo-to-multiview conversion for autostereoscopic displays. In: Computer graphics forum, vol. 33, pp. 63–72. Wiley, New York (2014)
6. Chaurasia, G., Duchêne, S., Sorkine-Hornung, O., Drettakis, G.: Depth synthesis and local warps for plausible image-based navigation. ACM Trans. Graph. **32**(3), 30:1–30:12 (2013)
7. Darsa, L., Costa, B., Varshney, A.: Navigating static environments using image-space simplification and morphing. In: Proceedings of the Symposium on Interactive 3D Graphics, pp. 25 – 34, 28–30 April 1997
8. Davidson, C.: Piku-Piku. www.start3d.com (2012). Accessed Nov 2012
9. Didyk, P., Ritschel, T., Eisemann, E., Myszkowski, K., Seidel, H.P.: Apparent stereo: the cornsweet illusion can enhance perceived depth. In: IS&T/SPIE electronic imaging, pp. 82,910N–82,910N. International Society for Optics and Photonics (2012)
10. Didyk, P., Ritschel, T., Eisemann, E., Myszkowski, K., Seidel, H.P., Matusik, W.: A luminance-contrast-aware disparity model and applications. ACM Trans. Graph. (TOG) **31**(6), 184 (2012)
11. Dosher, B.A., Landy, M.S., Sperling, G.: Kinetic depth effect and optic flow-I. 3D shape from Fourier motion. Vis. Res. **29**, 1789–1813 (1989)
12. Durgin, F.H., Proffitt, D.R., Olson, T.J., Reinke, K.S.: Comparing depth from motion with depth from binocular disparity. J. Exp. Psychol.-Hum. Percept. Perform. **21**, 679–699 (1995)
13. Epstein, W.: Perceptual invariance in the kinetic depth-effect. Am. J. Psychol. **78**(2), 301–303 (1965)
14. Gibson, E.J., Gibson, J.J., Smith, O.W., Flock, H.: Motion parallax as a determinant of perceived depth. J. Exp. Psychol. **58**(1), 40–51 (1959)
15. Gopi, M., Krishnan, S.: A fast and efficient projection-based approach for surface reconstruction. In: Computer graphics and image Processing, 2002. Proceedings. XV Brazilian Symposium on, pp. 179–186 (2002)
16. Harel, J., Koch, C., Perona, P.: Graph-based visual saliency. In: Advances in Neural Information Processing Systems 19, pp. 545–552. MIT Press, Cambridge (2007)
17. Hartley, R., Zisserman, A.: Multiple view geometry in computer vision. Cambridge University Press (2003)
18. Heineman, J.: Stereogranimator. http://stereo.nypl.org/ (2012). Accessed Nov 2012
19. Hoffman, D.M., Girshick, A.R., Akeley, K., Banks, M.S.: Vergence-accommodation conflicts hinder visual performance and cause visual fatigue. J. Vis. **8**, 33 (2008)
20. Ip, C.Y., Varshney, A.: Saliency-assisted navigation of very large landscape images. Vis Comput Graph. IEEE Trans. **17**(12), 1737–1746 (2011)
21. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. PAMI, IEEE Trans. **20**(11), 1254–1259 (1998)
22. Kellnhofer, P., Ritschel, T., Myszkowski, K., Seidel, H.P.: Optimizing disparity for motion in depth. In: Computer Graphics Forum, vol. 32, pp. 143–152. Wiley, New york (2013)
23. Kim, Y., Varshney, A.: Saliency-guided enhancement for volume visualization. IEEE Trans. Vis. Comp. Graph. **12**(5), 925–932 (2006)
24. Kim, Y., Varshney, A., Jacobs, D.W., Guimbretière, F.: Mesh saliency and human eye fixations. ACM Trans. Appl. Percept. **7**(2), 1–13 (2010). doi:10.1145/1670671.1670676
25. Krähenbühl, P., Lang, M., Hornung, A., Gross, M.: A system for retargeting of streaming video. ACM Trans. Graph. **28**(5), 126:1–126:10 (2009)
26. Landy, M.S., Dosher, B.A., Sperling, G., Perkins, M.E.: The kinetic depth effect and optic flow-II. First- and second-order motion. Vision Research **31**(5), 859–876 (1991)
27. Lang, M., Hornung, A., Wang, O., Poulakos, S., Smolic, A., Gross, M.H.: Nonlinear disparity mapping for stereoscopic 3D. ACM Transactions on Graphics (TOG) **29**(4) (2010)
28. Lee, C.H., Kim, Y., Varshney, A.: Saliency-guided lighting. IEICE Trans. Inf. Syst. **E92–D**(2), 369–373 (2009)
29. Lee, C.H., Varshney, A., Jacobs, D.: Mesh saliency. ACM Trans. Graph. (Proc. SIGGRAPH 2005) **24**(3), 659–666 (2005)
30. Lee, S., Kim, Y., Lee, J., Kim, K., Lee, K., Noh, J.: Depth manipulation using disparity histogram analysis for stereoscopic 3d. Vis. Comp. **30**(4), 455–465 (2014). doi:10.1007/s00371-013-0868-3
31. Liu, W., Wu, Y., Guo, F., Hu, Z.: An efficient approach for 2D to 3D video conversion based on structure from motion. Vis. Comp. **31**(1), 55–68 (2015). doi:10.1007/s00371-013-0904-3
32. Lytro: Perspective shift. www.lytro.com/camera#perspective_shift (2013). Accessed Dec 2013
33. Mark, W.R., McMillan, L., Bishop, G.: Post-rendering 3D warping. In: Proceedings of the 1997 symposium on Interactive 3D graphics, I3D '97, pp. 7–10. ACM, New York, USA (1997)
34. Michailidis, G.T., Pajarola, R., Andreadis, I.: High Performance stereo system for dense 3-D reconstruction. Circuits and system video for technology. IEEE Trans. **24**(6), 929–941 (2014)
35. Nagata, S.: How to reinforce perception of depth in single two-dimensional pictures. In: Stephen, R.E. (ed.) Pictorial communication in virtual and real environments, pp. 527–545. Taylor & Francis, Bristol (1991)
36. Nakayama, K., Tyler, C.W.: Psychophysical isolation of movement sensitivity by removal of familiar position cues. Vis. Res. **21**, 427–433 (1981)
37. Ono, M.E., Rivest, J., Ono, H.: Depth perception as a function of motion parallax and absolute-distance information. J Exp. Psychol.-Hum. Percept. Perform. **12**, 331–337 (1986)
38. Patro, R., Ip, C.Y., Bista, S., Varshney, A.: Social snapshot: a system for temporally coupled social photography. IEEE Comput. Graph. Appl. **31**, 74–84 (2011)
39. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Trans. Graph. **22**, 313–318 (2003)
40. Policarpo, F., Oliveira, M.M., Comba, J.L.D.: Real-time relief mapping on arbitrary polygonal surfaces. In: Proceedings of the 2005

Symposium on Interactive 3D Graphics and Games. I3D '05, pp. 155–162. ACM, New York, USA (2005)
41. Proffitt, D.R., Rock, I., Hecht, H., Schubert, J.: Stereokinetic effect and its relation to the kinetic depth effect. J. Exp. Psychol.-Hum. Percept. Perform. **18**, 3–21 (1992)
42. Robinson, D., Gordon, J., Gordon, S.: A model of the smooth pursuit eye movement system. Biol. Cybern. **55**, 43–57 (1986)
43. Rogers, B., Graham, M.: Motion parallax as an independent cue for depth perception. Perception **8**, 125–134 (1979)
44. Rogers, B., Graham, M.: Similarities between motion parallax and stereopsis in human depth perception. Vis. Res. **22**, 261–270 (1982)
45. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3D. In: SIGGRAPH Conference Proceedings, pp. 835–846. ACM Press, New York, USA (2006)
46. Sperling, G., Landy, M.S., Dosher, B.A., Perkins, M.E.: Kinetic depth effect and identification of shape. J. Exp. Psychol.-Hum. Percept. Perform. **15**, 826–840 (1989)
47. Stereographics: The Stereographics Developer's Handbook - Background on Creating Images for CrystalEyes and SimulEyes. Stereographics Corporation. http://www.reald-corporate.com/scientific/downloads/handbook.pdf (1997). Accessed Dec 2013
48. Sun, D., Roth, S., Black, M.: Secrets of optical flow estimation and their principles. In: Computer vision and pattern recognition (CVPR), 2010 IEEE Conference on, pp. 2432 –2439 (2010)
49. Toyoura, M., Kashiwagi, K., Sugiura, A., Mao, X.: Mono-glass for providing distance information for people losing sight in one eye. In: Proceedings of the 11th ACM SIGGRAPH international conference on virtual-reality continuum and its applications in Industry, pp. 39–42. ACM (2012)
50. Ujike, H., Yokoi, T., Saida, S.: Effects of virtual body motion on visually-induced motion sickness. In: Annual International Conference of the IEEE Engineering in Medicine and Biology Society (2004)
51. US. Dept. of Labor: Occuptional saftey and health administartion. www.osha.gov/SLTC/etools/computerworkstations/components_monitors.html (2013). Accessed Dec 2013
52. Vishwanath, D., Hibbard, P.B.: Seeing in 3-D with just one eye: stereopsis without binocular vision. Psychol. Sci. **24**(9), 1673–1685 (2013)
53. Wallach, H., O'Connell, D.N.: The kinetic depth effect. J. Exp. Psychol. **45**, 205–217 (1953)
54. Weyrich, T., Deng, J., Barnes, C., Rusinkiewicz, S., Finkelstein, A.: Digital bas-relief from 3D scenes. ACM Trans. Graph. **26**(3), 32 (2007)
55. Wilson, M.: Shooting challenge: Wiggle 3D. www.gizmodo.com/5895289/shooting-challenge-wiggle-3d (2012). Accessed Sept 2012
56. Yoonessi, A., Baker, C.: Contribution of motion parallax to depth ordering, depth magnitude and segmentation. J. Vis. **10**, 1194–1194 (2011)
57. Yoshida, K., Takahashi, S., Ono, H., Fujishiro, I., Okada, M.: Perceptually-guided design of nonperspectives through pictorial depth cues. In: Computer graphics, imaging and visualization (CGIV), 2010 Seventh International Conference on, pp. 173–178. IEEE (2010)
58. Zhang, C., Li, Z., Cheng, Y., Cai, R., Chao, H., Rui, Y.: Meshstereo: A global stereo model with mesh alignment regularization for view interpolation. In: International Conference on Computer Vision (2015)
59. Zheng, K.C., Colburn, R.A., Agarwala, A., Agrawala, M., Salesin, D., Curless, B.L., Cohen, M.F.: Parallax photography: creating 3D cinematic effects from stills. In: Proceedings of Graphics Interface, pp. 111–118. Canadian Information Processing Society (2009)
60. Zhu, C., Leow, W.: Textured mesh surface reconstruction of large buildings with multi-view stereo. Vis. Comput. **29**(6–8), 609–615 (2013). doi:10.1007/s00371-013-0827-z
61. Zitnick, C.L., Jojic, N., Kang, S.B.: Consistent segmentation for optical flow estimation. Int. Conf. Comput. Vis. **2**, 1308–1315 (2005)



Sujal Bista is a graphics researcher and a game developer. His research interests include computer graphics, virtual and augmented reality, scientific visualization, artificial intelligence, and human-motion understanding. Bista did his Ph.D. and postdoctoral research at the University of Maryland, College Park.

Ícaro Lins Leitão da Cunha is an assistant professor at the Universidade Federal Rural de Pernambuco. His research interests include computer graphics, virtual reality and computational mathematics. Cunha has received his bachelor's degree in computer science from the Universidade Federal da Paraíba, an M.S. degree in computational mathematics from the Universidade de São Paulo and a Ph.D. in computer engineering from the Universidade Federal do Rio Grande do Norte.

Amitabh Varshney is the Director of the Institute for Advanced Computer Studies (UMIACS), Professor of Computer Science at the University of Maryland at College Park, and Co-Director of the Center for Health-related Informatics and Bioimaging. Varshney's research focus is on exploring the applications of high-performance computing and visualization in engineering, science, and medicine. He has worked on a number of research areas including visual saliency, summarization of large visual datasets, and visual computing for big data. He is currently exploring general-purpose high-performance parallel computing using clusters of CPUs and graphics processing units (GPUs). He has served in various roles in the IEEE Visualization and Graphics Technical Committee, including as its Chair, 2008–2012. He received the IEEE Visualization Technical Achievement Award in 2004. He is a Fellow of IEEE.