Satish Kumar Singh
Partha Roy
Balasubramanian Raman
P. Nagabhushan (Eds.)

Communications in Computer and Information Science 1378

Computer Vision and Image Processing

5th International Conference, CVIP 2020
Prayagraj, India, December 4–6, 2020
Revised Selected Papers, Part III

Part 3

Springer

Communications in Computer and Information Science 1378

Editorial Board Members

Joaquim Filipe
Polytechnic Institute of Setúbal, Setúbal, Portugal

Ashish Ghosh
Indian Statistical Institute, Kolkata, India

Raquel Oliveira Prates
Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil

Lizhu Zhou
Tsinghua University, Beijing, China

Graph-Based Depth Estimation in a Monocular Image Using Constrained 3D Wireframe Models

Bishshoy Das, H. Pallab Jyoti Dutta(✉), and M. K. Bhuyan

Department of Electronics and Electrical Engineering, IIT Guwahati, Guwahati 781039, India
{bishshoy,h18,mkb}@iitg.ac.in

Abstract. In this paper, the problem of estimating the depth of an object from its monocular image is addressed. Here, basically, an algorithm is developed, which performs shape matching, and as a result, achieve accurate depth maps of objects. In the algorithm, first, an optimal camera position is determined. Then, the 3D model is projected onto the image plane of the camera, yielding a projected 2D image of the 3D model. An objective function determines a score based on graph-based feature matching, and the depth map is extracted from the geometrical information of the 3D model. Finally, the depth map and the original image are combined to create a 3D point cloud simulation of the object. Experimental analysis shows the efficacy of the proposed method.

Keywords: Monocular image · Depth map estimation · Wireframe model · 3D reconstruction

1 Introduction

Depth estimation of a scene from monocular images is an elemental problem in computer vision. With the rapid development of machine learning, most of the recent techniques of depth estimation have focused on learning depth estimator models from pixel data [1], stereo cues [2] and semantic labels [3]. Markov Random Fields trained via supervised learning of depth cues has ramified Make3D [4], which yields excellent depth maps of unstructured scenes. These techniques are well suited for outdoor scenes, and also where there are lots of pixels belonging to separate object classes, *viz.,* sky, tree, road, grass, water, building, mountain and foreground objects [3]. Other works rely on stereo vision [5,11] or structure derived from motion [6]. Both of these methods require two or more images. Shape from shading [7] extracts photometric cues from a single image, which becomes challenging when surfaces have non-uniform color or texture. Geometric cues like vanishing points [8], horizon and surface boundaries [9] can only determine the affine structure of images.

In this paper, we present an algorithm to generate depth maps only from a single image that contains an object on a white background, using graph-based

© Springer Nature Singapore Pte Ltd. 2021
S. K. Singh et al. (Eds.): CVIP 2020, CCIS 1378, pp. 419–433, 2021.
https://doi.org/10.1007/978-981-16-1103-2_35

image representation [10]. We focus on the very basic fact that an image is a projection of a 3D object on two dimensions of the image plane of a camera. The human brain assumes that the image of a hand-drawn cube is that of a cube because a 3D model of a cube exists well-defined in its memory. We make this assumption as the only explicit assumption in our algorithm. The proposed method is explained in the sections to follow.

2 Proposed Method

The algorithm flow is presented in Fig. 1. There are three computational blocks and one flowchart connecting all these blocks. In the graph feature extractor block, we take an image, use its corners and edges to determine a vertex list and an adjacency matrix of the graph representation of the image. In the score function block, we take two graph representations of two different images (the object image and the rendered image), and compute the distance between their representations in terms of the centroid, surface area, overlap area, vertex and edge scores. All these scores are then added together to form a combined score. In the optimization stage, we show the flow of the optimization algorithm. Finally, in the flowchart, we show how all the blocks are connected together.

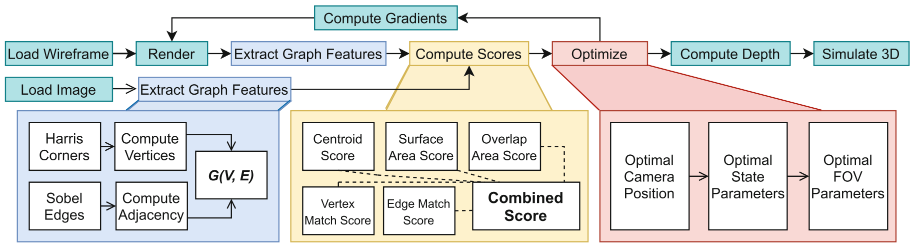

Fig. 1. Block diagram of the purposed algorithm

First, we load the object's monocular image. Then we extract its graph features using the graph feature extractor block. We also initiate a 3D model and feed it to the optimizer. The optimizer instantiates a state vector of the 3D model, feeds it to the renderer, which generates the image of the 3D model for the given state. Then the graph feature extractor extracts graph features from it. Both the graph representation of the object image and the rendered image of the 3D model is fed to the scoring function block, which computes all the different aforementioned scores and feeds it back to the optimizer. The optimizer performs gradient ascent and tweaks the state vector in a way to maximize the scores. After enough iterations, the optimal state vector is obtained which is used to extract the depth map and subsequently perform a point cloud simulation of the input object.

2.1 Graph-Based Features

Graph-based methods are used for structure detection, which also gives us an estimation of the depth map.

Graph Vertices: We set the corners of the image to be the vertices V of the graph G. The corners of the 2D object image (I_R-image) are extracted using the Harris corner detector.

Graph Adjacency Matrix: Adjacency matrix A_R is an $N \times N$ matrix, where $N = |V|$ and $A_R(i, j) = 0$ or $1, \forall\ 1 \leq i \leq N,\ 1 \leq j \leq N$ indicating the absence or presence of an edge, respectively, between v_i and v_j. To determine the values of the adjacency matrix A_R, we used the following approach. We extract the edge map image I_E of I_R using the Canny edge detector. We define a square mask image I_M of size $n \times n$ pixels. To determine $A_R(i, j)$, we center the mask on a corner point v_i. We traverse the mask I_M on a line connecting corners points v_i and v_j. We count the number of white pixels between them. If it is greater than 90% of the Manhattan distance between the two, then $A_R(i, j)$ is 1 or else 0.

2.2 Constrained 3D Wireframe Models

3D models of primitive shapes like cubes and spheres are created, adhering to the *OpenGL* format (homogeneous coordinate system). The vertices of the 3D model V and the edges E is defined as a graph $G = (V, E)$ in terms of adjacency matrix A. Centered around the origin, we allow the models to scale unconstrained in x, y and z directions. This consideration also allows higher level shapes to be taken into account using one single primitive model. A model of a cube may generate a family of cuboids. This generalization allows one model to encompass a larger number of 3D models while not explicitly changing the graph (wireframe). This imposes constraints on the 3D models and it reduces the number of possible solutions in the degenerate problem of finding a 3D model from a 2D image.

2.3 Operations on the 3D Models

Model-View-Projection Matrices: Our models are defined within the clip space specified by *OpenGL* specifications. So, the model matrix M is simply the identity matrix I_4. View matrix Λ models the camera's position $\overrightarrow{e}$, it's *look-at* direction $\overrightarrow{l}$ (from point $\overrightarrow{e}$) and it's orientation $\overrightarrow{u}$ (orthogonal to $\overrightarrow{l}$). In Sect. 2.6, we discuss our approach to find the optimal camera position. Direction at which the camera is pointing is fixed at $\overrightarrow{l} = [0\ 0\ 0]^T$, and orientation of the camera is fixed at $\overrightarrow{u} = [0\ 1\ 0]^T$ to reduce redundant camera movements. All the possible states of the wireframe model which can be generated by varying $\overrightarrow{l}$ and $\overrightarrow{u}$. These states can also be generated by varying $\overrightarrow{\theta}$. The projection matrix Pr models the field of view θ_{fov} of the *"lens"* mounted on the camera. We set the frustum's near plane (or the image plane) to be 0.1 unit from the camera and the far infinite plane to be 10 units away. θ_{fov} will be all also be optimized during gradient ascent. The above matrices are generated using the *OpenGL Mathematics* library. The model-view-projection matrix is therefore, can be represented as: $MVP(\overrightarrow{e}) = Pr \times \Lambda(\overrightarrow{e}) \times M = Pr \times \Lambda(\overrightarrow{e})$.

Affine Transformation: Affine Transformation is also applied in the order: Translation → Rotation → Scaling.

Thus, the Affine Transform matrix is defined as:

$$AT = \begin{pmatrix} s_x \cos\theta_y \cos\theta_z & -s_y \sin\theta_z \cos\theta_y & -s_z \sin\theta_y & t_x \\ \begin{array}{l} s_x \sin\theta_x \sin\theta_y \cos\theta_z \\ +s_x \cos\theta_x \sin\theta_z \end{array} & \begin{array}{l} -s_y \sin\theta_x \sin\theta_y \sin\theta_z \\ +s_y \cos\theta_x \cos\theta_z \end{array} & s_z \sin\theta_x \cos\theta_y & t_y \\ \begin{array}{l} s_x \cos\theta_x \sin\theta_y \cos\theta_z \\ -s_x \sin\theta_x \sin\theta_z \end{array} & \begin{array}{l} -s_y \cos\theta_x \sin\theta_y \sin\theta_z \\ -s_y \sin\theta_x \cos\theta_z \end{array} & s_z \cos\theta_x \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Note that the translation in z direction is redundant in the presence of scaling and hence eliminated. Finally the operations are combined as: $MVP \times AT$.

2.4 State Vector

The proposed state vector $\overrightarrow{X}$ is a *9-dimensional* vector, and it is represented as: $\overrightarrow{X} = [s_x\ s_y\ s_z\ \theta_x\ \theta_y\ \theta_z\ t_x\ t_y\ \theta_{fov}]^T$.

Projected Images: Each state vector $\overrightarrow{X}$ represents a state of the 3D model (first *8-dimensions*) and the camera parameter (θ_{fov}). Each state is rendered using the *OpenGL* API to generate the projected image I_X, which has a resolution of $R_X \times R_Y \times 3$ and 24-bit color depth. This projection operation is represented as: $P(\overrightarrow{X}) = I_X$. The corners of the 3D model are rendered as a square of 5 px width and blue color. Edges are rendered as white straight lines. Triangles are filled with red color, and the background is painted as black. In our tests, R_X =720 px and R_Y =720 px for all the images.

2.5 Score Functions

The score function $F(I_X)$ (or the combined score function), generates a numerical value based on the graph of $G_X = (V_X, E_X)$, (which is the graph of I_X) and other several sub-score functions. They are- Centroid score function $F_G(I_X)$, Vertex score function $F_V(V_X)$, Edge score function $F_E(E_X)$, Surface Area score function $F_S(I_X)$, Overlap Area score function $F_O(I_X)$.

Centroid Score: The centroid area score measures the closeness of the centroid of the two images, I_R and I_X. We define a function $L_X\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)$ on image I_X and pixel coordinate $[x\ \ y]^T$ as: $L_X\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{cases} 0, \bigwedge_{k=1}^{3} I_X(x,y,k) = 0, \begin{cases} 1 \le x \le R_X \\ 1 \le y \le R_Y \end{cases} \\ 1, \mathit{otherwise} \end{cases}$.

Also, we define $L_R\left(\begin{bmatrix} x \\ y \end{bmatrix}\right)$ on image I_R and pixel coordinate $[x\ \ y]^T$ as: $L_R\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{cases} 0, & \bigwedge_{k=1}^{3} I_R(x,y,k) = 255, \\ 1, & otherwise \end{cases} \begin{cases} 1 \le x \le R_X \\ 1 \le y \le R_Y \end{cases}$. L_X indicates whether $I_X(x,y)$ is foreground pixel or background pixel. All background pixels in I_X are *black*. The same is performed by L_R except for all the background pixels in I_R which are *white*.

The centroids of I_X and I_R are computed respectively as:

$$C(I_X) = \left\lfloor \frac{\sum_{i=1}^{R_X}\sum_{j=1}^{R_Y} L_X\left(\begin{bmatrix} i \\ j \end{bmatrix}\right)\begin{bmatrix} i \\ j \end{bmatrix}}{\sum_{i=1}^{R_X}\sum_{j=1}^{R_Y} L_X\left(\begin{bmatrix} i \\ j \end{bmatrix}\right)} \right\rceil, C(I_R) = \left\lfloor \frac{\sum_{i=1}^{R_X}\sum_{j=1}^{R_Y} L_R\left(\begin{bmatrix} i \\ j \end{bmatrix}\right)\begin{bmatrix} i \\ j \end{bmatrix}}{\sum_{i=1}^{R_X}\sum_{j=1}^{R_Y} L_R\left(\begin{bmatrix} i \\ j \end{bmatrix}\right)} \right\rceil$$

Next, we define $C'(I_R)$ as the maximum possible distance from the centroid of I_R and the square bounded by $[0\ \ 0]^T$ and $[R_X\ \ R_Y]^T$. We use this value for normalizing the centroid score.

$$C'(I_R) = \max\begin{pmatrix} \left\|C(I_R) - [0\ 0]^T\right\|_2, \\ \left\|C(I_R) - [R_X\ 0]^T\right\|_2, \\ \left\|C(I_R) - [0\ R_Y]^T\right\|_2, \\ \left\|C(I_R) - [R_X\ R_Y]^T\right\|_2 \end{pmatrix}$$

The centroid score of I_X is calculated as: $F_G(I_X) = 1 - \frac{||C(I_X) - C(I_R)||_2}{C'(I_R)}$

Graph-Based Scores: Graph-based scores enable the optimization algorithm to select structurally, better-projected images. We compare the graph representation of the object image I_R (which is $G_R = (V_R, E_R)$) with the graph representation of the projected image I_X (which is $G_X = (V_X, I_X)$) to generate two sets on independent scores, the vertex score $F_V(V_X)$ and the edge score $F_E(V_E)$.

Vertex Score: Corners of the image I_X are set as the vertices of the graph representing I_X. Corners of I_X are detected by detecting patches of blue pixels and the coordinates are set as V_X. To compute the vertex score, we associate all vertices in V_X with some vertex in G_R (which is the graph for I_R, *i.e.,* $G_R = (V_R, E_R)$) and form a new vertex list, V_C. The i^{th} item in the list V_C is defined as:

$$V_C^{(i)} = \underset{v\ \in\ V_R}{\arg\min} \left\|V_X^{(i)} - v\right\|_2$$

$V_C^{(i)}$ is, therefore, that vertex in G_R, which is closest to the vertex $V_X^{(i)}$; the vertex distance being measured as the Euclidean distance between the two pixels they represent. The proposed vertex score is then computed as:

$$F_V(V_X) = 1 - \frac{\sum_{i=1}^{|V_X|} \|V_X(i) - V_C(i)\|_2}{|V_X|\ \sqrt{R_X^2 + R_Y^2}}$$

Edge Score: The edge score measures the number of edges common in both graphs G_R and G_X. Calculation of *Edge score* is illustrated in Fig. 2.

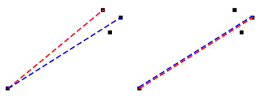

Fig. 2. (a) Blue dotted line indicates an edge in G_R. Red dotted line indicates an edge in G_X. The corners are matched but edges are not. *Corner score* = 1*, Edge score* = 1. (b) Blue dotted line indicates an edge in G_R. Red dotted line indicates an edge in G_X. Both corners and edges are matched. *Corner score* = 1*, Edge score* = 2 (Color figure online)

On a global scale, a higher edge score means a better state vector of the 3D model. For example, in Fig. 3 both the configuration $(3.b)$ and $(3.c)$ yield very high centroid scores, surface area scores and overlap area scores, but $(3.c)$ gives less edge score. Thus, structurally, configuration $(3.c)$ is preferred over the configuration $(3.b)$, in the optimization process. We use the algorithm used for Graph Adjacency Matrix to construct the adjacency matrix A_X of G_X (graph of I_X). Hereafter, we compare the adjacency matrices of the two graphs (V_C, A_X) and (V_R, A_R) to calculate the edge score by exploiting the fact that $V_C \subseteq V_R$. We create a list λ, the i^{th} element of which is given by: $\lambda^{(i)} = j \;\mid\; V_C^{(i)} = V_R^{(j)}$.

Therefore, the mapping λ indicates the location in V_R at which a vertex in V_C is to be found out. Consequently, we can also get to know which row-column in A_X should be compared to row-column in A_R for an edge in A_X to be "matched" with an edge in A_R. Then, we count the number of edges "matched" in both the adjacency matrices. To avoid over-counting, we initialize a matrix ρ of dimensions $|V_R| \times |V_R|$ and initialize all the values in ρ as 0. We mark $\rho(\lambda(i), \lambda(j))$ to 1 to indicate that the edge $A_R(\lambda(i), \lambda(j))$ has been counted. Since A is symmetric, we also mark $\rho(\lambda(j), \lambda(i))$ as counted. We initialize a counting variable, $c = 0$, and apply the following formula to count the number of edge matches as:

$$\left.\begin{array}{c} c := c + 1 \\ \rho(\lambda(i), \lambda(j)) = 1 \\ \rho(\lambda(j), \lambda(i)) = 1 \end{array}\right\}, \text{ if } \left\{\begin{array}{c} A_X(i, j) = 1 \\ A_R(\lambda(i), \lambda(j)) = 1 \\ \rho(\lambda(i), \lambda(j)) = 0 \end{array}\right.$$

We formulate our proposed edge score as: $F_E(E_X) = 1 + c$

The addition of 1 doesn't allow the combined score to become zero if there are absolutely no edge matches, which may happen during the initial few iterations of optimization.

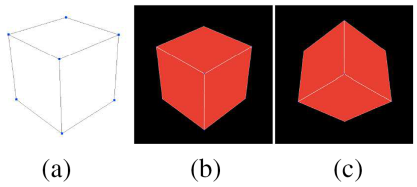

Fig. 3. (a) A graph of a cube object; (b) A configuration of a 3D cube model. *Edge score* = 10, since there are 9 matched edges with the graph shown in (a). The *centroid score, surface area score* and *overlap area scores* are very high; (c) A different configuration, which also has a very high *centroid score, surface area score* and *overlap area score*, but has a lower *Edge score* of 7, since there are only 6 matched edges.

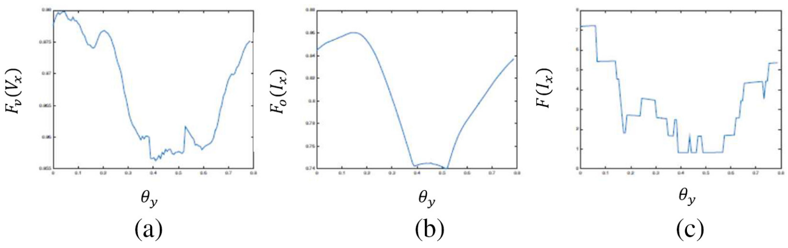

Fig. 4. Variation of different score functions in the optimization process of fitting a 3D wireframe model of a cube in a similar cube like 2D image (in this case, a Rubik's cube). All state parameters are determined at optimization stage 1, where a good camera position is obtained. Then the cube is rotated about the y-axis, and the following plots of various scores are noted. (a) Plot of $F_V(V_X)$ vs. θ_y ; (b) Plot of $F_O(I_X)$ vs. θ_y; (c) Plot of $F(I_X)$ vs. θ_y. This indicates that the overlap area score is a smoother function, and hence, it is well applicable for gradient ascent.

Surface Area Score: The surface area score measures the area footprint of I_X and compares to that of I_R. The surface area σ is calculated as:

$$\sigma(I_R) = \sum_{i=1}^{R_X} \sum_{j=1}^{R_Y} L_R\left(\begin{bmatrix} i \\ j \end{bmatrix}\right), \sigma(I_X) = \sum_{i=1}^{R_X} \sum_{j=1}^{R_Y} L_X\left(\begin{bmatrix} i \\ j \end{bmatrix}\right)$$

The proposed surface area score is defined as: $F_S(I_X) = 1 - \frac{|\sigma(I_X) - \sigma(I_R)|}{\sigma(I_R)}$

Overlap Area Score: The pixels common in I_R and I_X is calculated as (Fig. 4):

$$L_{X,R}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{cases} 1, L_X\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = L_R\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) \\ 0 \qquad\qquad otherwise \end{cases}$$

The overlap area $\Omega(I_X)$ is defined as: $\Omega(I_X) = \sum_{i=1}^{R_X} \sum_{j=1}^{R_Y} L_{X,R}\left(\begin{bmatrix} i \\ j \end{bmatrix}\right)$
The union area $\Gamma(I_X)$ is defined as: $\Gamma(I_X) = \sigma(I_R) + \sigma(I_X) - \Omega(I_X)$
The overlap area score is measured as: $F_O(I_X) = 1 - \frac{\Gamma(I_X) - \Omega(I_X)}{\Gamma(I_X)}$

Combined Score: The proposed combined score $F(I_X)$ is defined such that the effect of both overlap area score and the edge score are high and is given as:

$$F(I_X) = F_E(E_X) \times \frac{c_1 F_G(I_X) + c_2 F_V(V_X) + c_3 F_S(I_X) + c_4 F_O(I_X)}{c_1 + c_2 + c_3 + c_4}$$

2.6 Proposed Optimization Stages

Stage 1: Finding Optimal Starting Point for Gradient Ascent

Stage 1.1: Optimal Translation Parameters. We begin by optimizing the centroid score $F_G(I_X)$ over the translation parameters t_x and t_y of the state vector $\overrightarrow{X}$ for better accuracy in the following progressive stages. t_x^* and t_y^* are the optimal translation parameters and the operation is performed as: $\{t_x^*, t_y^*\} = \arg\max_{t_x,\ t_y} F_G(P(\overrightarrow{X})),$ subject to: $t_x \in \mathbb{R},\ t_y \in \mathbb{R}$
The update equation is defined as: $\overrightarrow{X} := \overrightarrow{X} + \eta \overrightarrow{\nabla} F_G(P(\overrightarrow{X}))$
The gradient operation is performed as:

$$\frac{\partial F_G(P(\overrightarrow{X}))}{\partial x_i} = \frac{F_G(P([x_1, \ldots, x_i + h, \ldots, x_9]^T)) - F_G(P(\overrightarrow{X}))}{h}$$

Stage 1.2: Initial Pass for Determining Scaling Parameters. This stage of the optimization process tries to obtain a good camera position for viewing the 3D model. We fix the camera at a distance of 20 units of clip space coordinates, invoke gradient ascent on the overlap area score, and restrict the search dimensions only to the scaling parameters $(s_x,\ s_y,\ s_z)$, such that $s_x = s_y = s_z$.

$$\arg\max_{s_x, s_y, s_z} F_O(P(\overrightarrow{X})), \text{subject to: } s_x \in \mathbb{R},\ s_y \in \mathbb{R},\ s_z \in \mathbb{R},\ s_x = s_y = s_z$$

The update equation is therefore be written as: $\overrightarrow{X} := \overrightarrow{X} + \eta \overrightarrow{\nabla} F_O(P(\overrightarrow{X}))$ The gradient operation is performed as:

$$\frac{\partial F_O(P(\overrightarrow{X}))}{\partial x_i} = \frac{F_O(P([x_1, ..., x_i + h, ..., x_9]^T)) - F_O(P(\overrightarrow{X}))}{h}$$

Based on the following criteria, η and h are adjusted with each iteration j. Let the mean combined score in the j^{th} iteration be:

$$\mu^{(j)} = \left(\frac{1}{9} \sum_{i=1}^{9} \frac{\partial F(P(\overrightarrow{X}))}{\partial x_i} \right) \Bigg|_j$$

Then η_{j+1} and h are calculated as:

$$\eta_{j+1} = \begin{cases} \eta_j/2, \text{ if } \mu_j > \mu_{j+1} \\ 0.1, \quad \text{if } \eta_j < 0.01 \end{cases}, \quad h := \frac{h}{2}, \text{ if } \eta_j < 0.01$$

Stage 1.3: Optimal Camera Position and Scaling Parameters. We place the camera in several latitude-longitude intersections of an imaginary sphere. Denoting, latitudes as ϕ and longitudes as θ, we have limits $-90° \leq \phi < 90°$ and $0° \leq \theta < 360°$ to cover the entire sphere. We generate linearly spaced latitudes in set Φ and linearly spaced latitudes in set Θ. So for every element in set $\Phi \times \Theta$, we compute camera position vector $\overrightarrow{p}$ as:

$$\overrightarrow{p} = \begin{bmatrix} r\cos(\theta)\cos(\phi) & r\sin(\theta)\cos(\phi) & r\sin(phi) \end{bmatrix}^T$$

We update the view matrix Λ with camera position $\overrightarrow{p}$, and optimize the overlap area score function with the same constraints over the scaling parameter. The gradient ascent update is then performed as:

$$\underset{s_x, s_y, s_z}{\arg\max} F_O(P(\overrightarrow{X})), \text{subject to: } s_x \in \mathbb{R},\ s_y \in \mathbb{R},\ s_z \in \mathbb{R},\ s_x = s_y = s_z$$

The update equation is: $\overrightarrow{X} := \overrightarrow{X} + \eta \overrightarrow{\nabla} F_O(P(\overrightarrow{X}))$
The optimal camera position $\overrightarrow{p^*}$ is that $\overrightarrow{p}$ which yields the highest combined score, and it is given by: $\overrightarrow{p^*} = \arg\max_{\overrightarrow{p}} F(P(\overrightarrow{X}) \mid \overrightarrow{p})$
The optimal scaling parameter s^* is therefore be written as:

$$s^* = \underset{s_x}{\arg\max} F(P(\overrightarrow{X}) \mid \overrightarrow{p^*}), \text{subject to: } s_x \in \mathbb{R}, s_y \in \mathbb{R}, s_z \in \mathbb{R}, s_x = s_y = s_z$$

Stage 2: Optimal Wireframe Model Parameters. After determining the optimal camera position $\overrightarrow{p^*}$ and the optimal scaling parameters s^*, we use this information to initiate gradient ascent on all the parameters of the state vector $\overrightarrow{X}$, except for the camera parameter θ_{fov}. As the first 8 parameters control the wireframe model alone, it would be computationally beneficial as optimization in one of the dimensions is excluded. We have chosen 30° or $\pi/6$ as the field of view of the camera's lens. To maximize the overlap area score, only 8 parameters of the wireframe model are taken into account. We optimize the overlap area score $F_O(I_X)$, but select the optimal state vector having the highest combined score $F(I_X)$. Thus, the optimization process and the update equations are defined as follows:

$$\underset{s_x, s_y, s_z, t_x, t_y, r_x, r_y, r_z}{\arg\max} F_O(P(\overrightarrow{X}) \mid \overrightarrow{p^*}) \text{ subject to: } s_x, s_y, s_z, t_x, t_y, r_x, r_y, r_z \in \mathbb{R}$$

$$\overrightarrow{X} := \overrightarrow{X} + \Psi \times \overrightarrow{\nabla} F_O(P(\overrightarrow{X}))$$

where, Ψ is a diagonal matrix. The i^{th} diagonal element $\Psi(i,i)$ represents the step size for the i^{th} dimension. We use the following approach to determine $\Psi(i,i)$.

$$\text{If } F_O(P(\overrightarrow{X})) < 0.9, \Psi(i,i) = \begin{cases} 0.1, & \text{if } \frac{\partial F_O(P(\overrightarrow{X}))}{\partial x_i} \geq \frac{1}{9}\sum\limits_{i=1}^{9} \frac{\partial F_O(P(\overrightarrow{X}))}{\partial x_i} \\ 0.05, & otherwise \end{cases}$$

$$\text{If } F_O(P(\overrightarrow{X})) \geq 0.9, \quad \Psi(i,i) = 0.05$$

Stage 3: Optimal Wireframe Model and Camera Model Parameters. In the final stage, we allow all the parameters of the state vector to be modified in order to improve the scores even further. Thus, we also find the optimal camera model parameter θ_{fov} in this stage. The optimization and update equations are:

$$\arg\max_{\overrightarrow{X}} F_O(P(\overrightarrow{X}) \mid \overrightarrow{p^*}) \text{ subject to: } s_x, s_y, s_z, t_x, t_y, r_x, r_y, r_z, \theta_{fov} \in \mathbb{R}$$

$$\overrightarrow{X} := \overrightarrow{X} + \eta \overrightarrow{\nabla} F_O(P(\overrightarrow{X}))$$

2.7 Depth Map Extraction

The actual depth values are normalized and all the non-object pixels are set to 0. The new depth values are: $D(i,j) = \begin{cases} \frac{d_{max} - Z(i,j)}{d_{max} - d_{min}}, & \text{if } Z(i,j) \neq 1 \\ 0 & \text{otherwise} \end{cases}$

2.8 Combining Object Image and Depth Map

After we extract the depth map D, we combine it with object image I_R to simulate a 3D experience of what the real object actually was.

3 Experimental Results

We tested our algorithm on various images of different structures. A comparison of the ground truth depth maps, the depth maps generated by our algorithm and other methods are shown in Fig. 5. Also, optimal camera positions obtained after optimization stage 1 are shown, along with the final wireframe model output. A comparison of a 2D image and its depth map is shown by overlaying the 2D image on the depth map with 50% transparency. The scores are shown in Table 2. The RMS error between the ground truth and the depth map generated by our algorithm is given in Table 1.

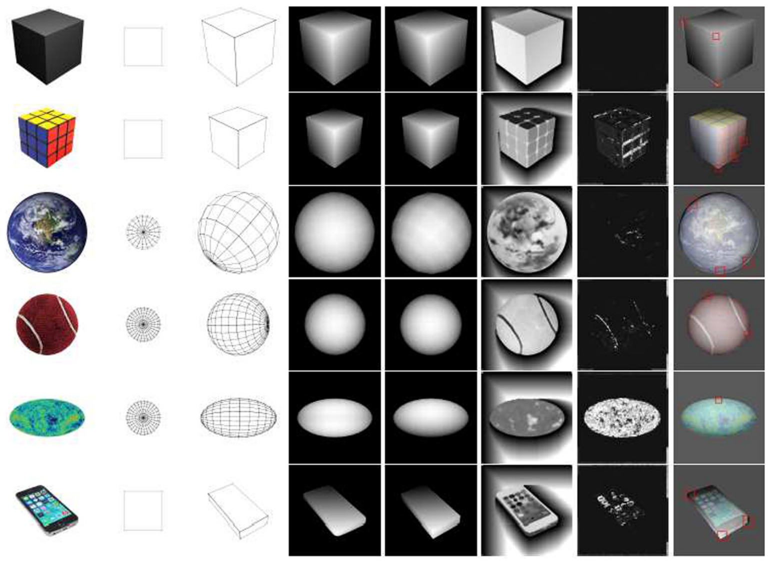

Fig. 5. Col 1 shows 2D object image; Col 2 shows corresponding 3D wireframe model and it's initial state; Col 3 shows final state of the wireframe model (i.e. output of optimization stage 3); Col 4 shows Ground truth depth map; Col 5 shows the depth map generated by our algorithm; Col 6 shows the depth map generated by shape from shading algorithm [12]; Col 7 shows the depth map generated by depth from defocus algorithm [13]; Col 8 shows overlay comparison of object and depth map. Red squared marks show erroneous areas. (Color figure online)

Table 1. RMS error between the ground truth and the generated depth map

Cube	Rubik's Cube	Earth	Tennis Ball	CMB Map	iPhone
2.892904	0.123181	3.420901	1.975758	5.801795	4.013128

Table 2. Comparison of 2D image and its depth map in terms of a matching score

Object image	Centroid score $F_G(I_X)$	Vertex score $F_V(V_X)$	Edge score $F_E(E_X)$	Surface area score $F_S(I_X)$	Overlap area score $F_O(I_X)$	Combined score $F(I_X)$
Cube	1	0.993406	10	0.998222	0.984951	9.89104
Rubik's Cube	1	0.993827	9	0.997169	0.983042	8.89238
Earth	0.983197	0.976024	107	0.999979	0.96473	103.92
Tennis Ball	0.998044	0.964645	369	0.999724	0.979267	360.614
CMB Map	0.998041	0.971926	404	0.99506	0.978407	395.369
iPhone	0.997234	0.992845	2	0.985505	0.955688	1.94163

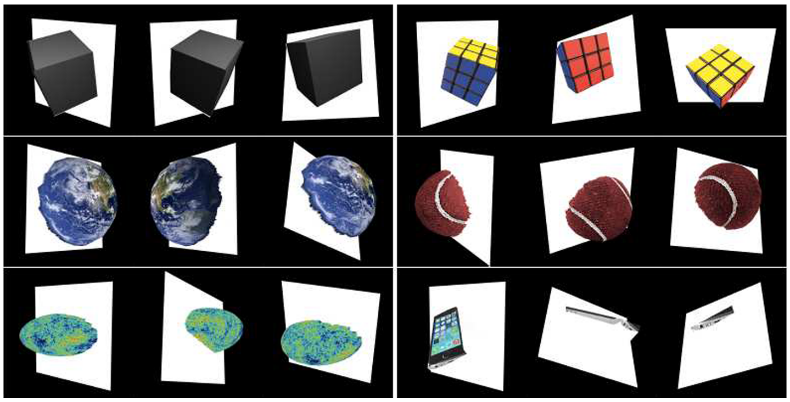

Fig. 6. 3D simulation of Cube, Rubik's Cube, Earth, Tennis Ball for 200 iterations each; CMB Map for 300 iterations; iPhone for 2500 iterations.

3.1 3D Simulation

The simulation of different images are shown in Fig. 6 for different angles. In Fig. 8, we show the transition of the 3D wireframe model from stage 1 (after optimal camera position has been computed) to the final configuration in the optimization stage 3. Even though the wireframe model gets close, but fails to improve after a certain point because of the rounded corners of the iPhone 5S. Figure 9 shows the plot of the RMS error of the depth map over 1200 iterations.

3.2 Facial Reconstruction

Our proposed algorithm can also be employed for 3D facial reconstruction. We demonstrate that using a generic wireframe model of a human face and images of human faces, the algorithm can be used to find the depth map of a human face. Thus, the image and the depth map can be combined to produce a reconstructed 3D model of a person's face (shown in Fig. 7). A part of the experimental simulation of our proposed 3D reconstruction algorithm is available at https://youtu.be/9sz1yvcGoBo.

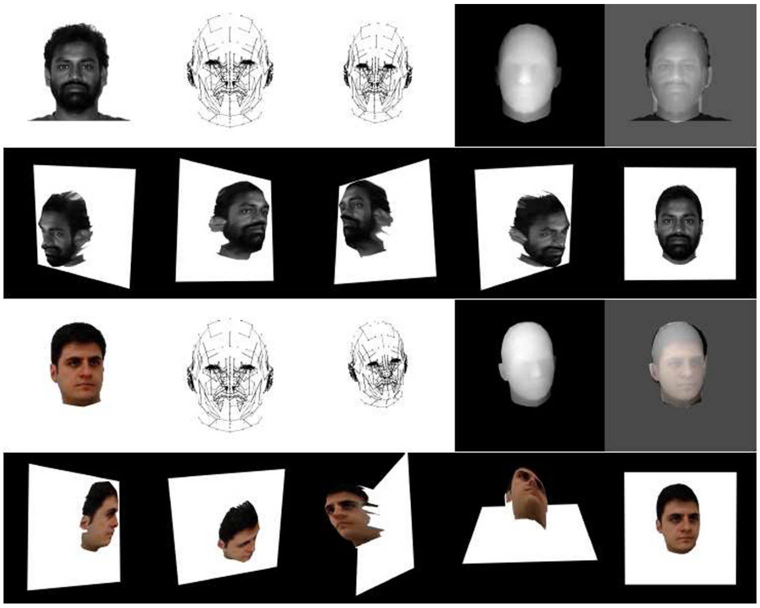
Fig. 7. Row 1 shows (a) Image from Yale Face Database; (b) Initial state of the face wireframe model; (c) Final state after optimization; (d) Depth map of the image of (a) Row 1; (e) Overlay of images (a) and (d) of Row 1. Row 2 shows 3D simulation of the images of Row 1. Row 3 shows (a) Image from FEI Face Database; (b) Initial state of the face wireframe model; (c) Final state after optimization; (d) Depth map of the image (a) of Row 3; (e) Overlay of (a) and (d) of Row 3. Row 4 shows 3D simulation of the images of Row 3.

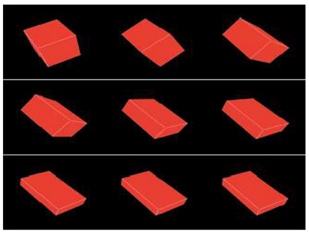
Fig. 8. States of the cube model during iterations of stage 3 optimization. Configuration of the wireframe model after (a) 0, (b) 100, (c) 200, (d) 300, (e) 500, (f) 600, (g) 800, (h) 1200, and (i) 2500 iterations.

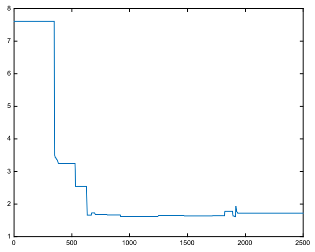

Fig. 9. Depth map RMS error vs. iteration in the estimation of iPhone.

4 Conclusion

When a human brain performs depth perception/estimation of a 2D image, it determines the 3D shape of the object from previous experience and the 3D model registered in its memory. In this paper, we tried to construct an optimization algorithm that mimics the “fitting” procedure followed by our brain for depth estimation. The advantage of this procedure is that it does not rely on shading, lighting, or stereo information. We present an algorithm to generate depth maps only from a single image. We considered the 3D model fitting of some basic shapes (cubes, spheres), and we also considered extended versions of these shapes (family of cubes/cuboids, the family of spheres/ellipsoids). However, our proposed algorithm can be extended and applied to more complicated shapes like face, tree, etc. as well. The limitation of our algorithm is that the objective function is non-convex, very jagged, with high value gradients throughout the search space. It is very difficult to converge at the global extrema. Also, another disadvantage of our algorithm is that if the object image has a structure not exactly as that of the 3D model, then the fitting will be bad and less than perfect.

References

1. Saxena, A., Chung, S.H., Ng, A.Y.: Learning depth from single monocular images. In: Advances in Neural Information Processing Systems, pp. 1161–1168 (2006)
2. Saxena, A., Schulte, J., Ng, A.Y.: Depth estimation using monocular and stereo cues. IJCAI **7**, 2197–2203 (2007)
3. Liu, B., Gould, S., Koller, D.: Single image depth estimation from predicted semantic labels. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1253–1260. IEEE (2010)
4. Saxena, A., Sun, M., Ng, A.Y.: Learning 3-D scene structure from a single still image. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8 (2007)
5. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**(1–3), 7–42 (2002). https://doi.org/10.1023/A:1014573219977
6. Forsyth, D., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall, Upper Saddle River (2011)
7. Zhang, R., Tsai, P.-S., Cryer, J.E., Shah, M.: Shape-from-shading: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **21**(8), 690–706 (1999)
8. Criminisi, A., Reid, I., Zisserman, A.: Single view metrology. Int. J. Comput. Vis. **40**(2), 123–148 (2000). https://doi.org/10.1023/A:1026598000963
9. Heitz, G., Gould, S., Saxena, A., Koller, D.: Cascaded classification models: combining models for holistic scene understanding. In: Advances in Neural Information Processing Systems, pp. 641–648 (2009)
10. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall Inc., Upper Saddle River (1988)
11. Malathi, T., Bhuyan, M.K.: Asymmetric occlusion detection using linear regression and weight-based filling for stereo disparity map estimation. IET Comput. Vis. **10**(7), 679–688 (2016). https://doi.org/10.1049/iet-cvi.2015.0214

12. Ping-Sing, T., Shah, M.: Shape from shading using linear approximation. Image Vis. Comput. **12**(8), 487–498 (1994)
13. Chakrabarti, A., Zickler, T.: Depth and deblurring from a spectrally-varying depth-of-field. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 648–661. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_47