⬅ ➡

FAST AND ACCURATE 3D SHAPE FROM FOCUS USING DYNAMIC PROGRAMMING OPTIMIZATION TECHNIQUE

Muhammad Bilal Ahmad and Tae-Sun Choi, SeniorMember IEEE

Gwangju Institute of Science and Technology, Gwangju, Korea. Email: bilal@gist.ac.kr

ABSTRACT

The conventional Shape from Focus (SFF) method for 3D shape recovery from image focus is fast but inaccurate. The SFF method based on the Focused Image Surface (FIS) has shown better results by exhaustive search of the FIS shape using planar surface approximation at the cost of considerably higher computations. In this paper, we investigate fast and accurate SFF method. The conventional SFF method is used as the rough estimate at pixels at regular steps in the x and y directions, and this rough estimate is used to search the FIS shape for all pixels between the steps using the Dynamic Programming optimization technique. The proposed algorithm is very fast and shows comparable results with those of accurate SFF methods.

1. INTRODUCTION

Shape From Focus (SFF) [1-4] for 3D shape recovery is a search method which searches the camera parameters (lens position and/or focal length) that correspond to focusing the object. The basic idea of image focus is that the objects at different distances from a lens are focused at different distances. In SFF, an unknown object is moved with respect to the imaging system and a sequence of images that correspond to different levels of object focus is obtained. A focus measure is computed in the small image regions of each of the image frame in the image sequence. The value of the focus measure increases as the image sharpness or contrast increases and it attains the maximum for the sharpest focused image. Thus the sharpest focused image regions can be detected and extracted. Further, the distance or depth of object surface patches that correspond to the small image regions can be obtained from the knowledge of the lens position and the focal length that result in the sharpest focused images of the surface patches.

The traditional SFF method (SFFTR) [2] maximizes the focus measure in planar images. SFFTR method is very fast but does not yield accurate shape or depth-map of objects because of piecewise constant approximation of the object shape in the window. The SFF method SFFFIS [3] based on the Focused Image Surface (FIS) yielded more accurate results than the SFFTR method. The FIS of an object is defined as the surface formed by the set of points at which the object points are focused by a camera lens. According to paraxial-geometric optics, there is one-to-one correspondence between the shape of an object and the shape of its FIS. Therefore the problem of shape recovery can be posed as the problem of determining the shape of the FIS. The SFFFIS method has increased the accuracy of 3D shape recovery of object surfaces, but at the cost of much higher computations.

The authors have already presented a new SFF technique SFFDP [4] to recover the 3D shape of objects using dynamic programming optimization technique. The search of FIS shape was presented as an optimization problem i.e. maximizes the focus measure in a 3D image volume. The SFFDP has low computational complexity and yields a better performance with respect to the 3D shape recovery. The SFFDP algorithm is significantly faster than the FIS algorithm, but a little slower than the SFFTR algorithm. In this paper, the SFFDP algorithm is modified to further reduce the number of computations and to keep the accuracy of 3D shape recovery.

2. THE PROPOSED ALGORITHM

In this paper and in [4], the search of the FIS shape is presented as an optimization problem i.e. maximizes the focus measure in the 3D image volume. The dynamic programming optimization technique is used to search the optimal shape of FIS, and hence, the recovery of the 3D shape. In SFFDP, the search of FIS was done in the whole image volume. To reduce the number of computations, here, we present the idea of searching the optimal surface in the small image volume around the first estimate by using the fast heuristic method based on DP. The optimal surfaces obtained from the small volumes are then combined together to get the final 3D shape.

In SFFDP_1, an input image sequence is considered as an image volume $V_{i,x,y}$, where x, y represent the number of rows and columns of each image frame respectively and i represents the number of images in the sequence. A focus measure operator, the Modified Laplacian [2], is applied on each image frame in the input image sequence, and get the focus measure image volume $O_{i,x,y}$. In the first phase, the rough estimate using the traditional method SFFTR is obtained at pixels with small steps in x and y directions. In the second phase, the rough estimate is used to search the depth map for pixels between the steps by searching for an FIS shape. For every window in which the FIS was estimated in the first step, a small cubic volume of image space is considered in the focus measure image volume. The volume is centered at the initial estimate of FIS in that window, and the center of x and y steps. Now, in this volume, a search is made using the proposed heuristic model of dynamic programming optimization technique to search for the closest match of the shape of FIS by maximizing the focus measure in that small volume. The whole image volume is divided into small volumes, and the FIS shape is searched in each image volume.

A small image volume $S_{p,m,n}$ centered with the image number from the rough estimate at (x,y) is taken from the focus measure image volume $O_{i,x,y}$. The size of $S_{p,m,n}$ is equal to the number of images chosen around the rough estimate and the size of steps in x and y directions during the first step for searching the rough estimate. Let P denotes the number of images in the small volume, and M and N denotes the step size in x (number of rows) and y (number of columns) directions. The values of P, M and N are arbitrary. Now the problem is reduced to search the optimal path in the small volume $S_{p,m,n}$ that maximizes the focus measure. The small volume is shown in Fig. 1 (a). It should be noted that the small volume is centered with midpoint of x and y steps. Now, 2D general networks are constructed from the 3D image volume $S_{p,m,n}$. The 2D networks are constructed by slicing $S_{p,m,n}$ along x and y directions. First, m is kept constant, and y-slice 2D networks $A^m_{p,n}$ are made. The y-slice 2D networks are made from one row of each image frame of $S_{p,m,n}$, where the row number is determined by the value of m. For example, $A^1_{p,n}$ is made from the first row of each image frame of $S_{p,m,n}$. Figure 1 (b) shows the y-slice 2D network for the first row. The number of y-slice networks is equal to the number of rows of image frame in $S_{p,m,n}$. Second, n is kept constant, and x-slice 2D networks $B^n_{p,m}$ are made. The x-slice 2D networks $B^n_{p,m}$ are made from one column of each image frame of $S_{p,m,n}$, where the column number is determined by the value of n. Figure 1 (c) shows the x-slice 2D network for the first column. The number of x-slice networks is equal to the number of columns of image frame in $S_{p,m,n}$. The total 2D networks are, therefore, equal to the size of each image frame in $S_{p,m,n}$.

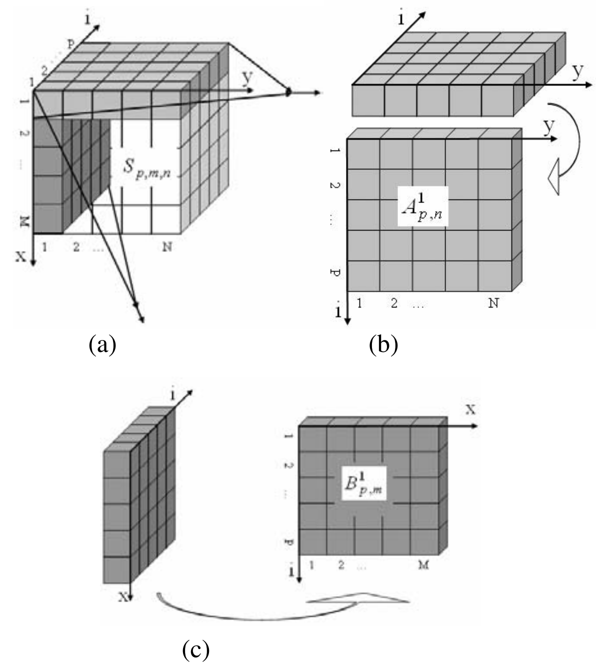


Fig. 1. (a) 3D small image volume $S_{p,m,n}$ around the rough estimate (b) 2D y-slice network for the first row (c) 2D x-slice network for the first column.

For small volume $S_{p,m,n}$, we have 'M' y-slice 2D networks of size P x N and 'N' x-slice 2D networks of size P x M. After slicing, "Right_Sum" and "Left_Sum" for each of the 2D networks are calculated using the recurrence formulation. The recurrence formulation for y-slice 2D networks is explained as follows. The same formulation can be easily applied for x-slice 2D networks. Consider the y-slice 2D network as shown in Fig. 1 (b). For simplicity, the superscript '1' is removed. So for 1st row of the small volume $S_{p,m,n}$, the y-slice 2D network (see Fig. 1 (b)) can be expressed in matrix form as:

$A = A^1_{p,n} = [a_{p,n}]$, where the size of matrix A is P x N .

The Right_Sum R^{Sum} for the matrix A is defined as:

$$R^{sum} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n-1} & r_{1,n} & \cdots & \cdots & r_{1,N} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n-1} & r_{2,n} & \cdots & \cdots & r_{2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \cdots & r_{p-1,n-1} & \vdots & \cdots & \vdots & \vdots \\ r_{p,1} & r_{p,2} & \cdots & r_{p,n-1} & r_{p,n} & \cdots & \cdots & r_{p,N} \\ \vdots & \vdots & \cdots & r_{p+1,n-1} & \vdots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ r_{P-1,1} & r_{P-1,2} & \cdots & r_{P-1,n-1} & r_{P-1,n} & \cdots & \cdots & r_{P-1,N} \\ r_{P,1} & r_{P,2} & \cdots & r_{P,n-1} & r_{P,n} & \cdots & \cdots & r_{P,N} \end{bmatrix}$$

where the recursive formulae involved in the calculation of the Right_Sum (see the "structure of matrix" R^{sum}) are given as:

$r_{p,1} = a_{p,1}$ for p = 1,2,…,P,

$r_{1,n} = a_{1,n} + \max\{r_{1,n-1}, r_{2,n-1}\}$,

$r_{p,n} = a_{p,n} + \max\{r_{p-1,n-1}, r_{p,n-1}, r_{p+1,n-1}\}$

for p = 2,3,…,P-1; n = 2,3,…,N, and

$r_{P,n} = a_{P,n} + \max\{r_{P-1,n-1}, r_{P,n-1}\}$, for n = 2,3,…,N.

In general, we can say that the Right_Sum at (p,n) is the sum of Laplacian value at (p,n) from the matrix A, and the maximum of the three previous Right_Sum values at the left column.

Similarly the Left_Sum L^{Sum} for the matrix A is defined. The recursive formulae involved in the calculation of Left_Sum is almost similar as for the Right_Sum R^{Sum}. It should be noted that the matrix Left_Sum L^{Sum} is filled from the right side, i.e., the last column of L^{Sum} is filled first and the first column is filled at the end. And, we can say that the Left_Sum at (p,n) is the sum of Laplacian value at (p,n) from the matrix A, and the maximum of the three previous Left_Sum values at the right column.

After calculating the Right_Sum and the Left_Sum for the matrix A, we calculate the "Total_Sum" T^{sum} for the matrix A as:

$$T^{sum} = R^{sum} + L^{sum} - A \,, \tag{1}$$

where $T^{sum} = [t_{p,n}]$, for p = 1,2,…,P; n = 1,2,…, N.

Similarly, the Total_Sum T^{sum} is calculated for all rows using (1). The superscript '1' removed from $A^{1}_{p,n}$ is introduced here as m, which represents the number of rows in $S_{p,m,n}$. So in general, the Total_Sum for y-slice networks can be rewritten as:

$$[YT^{sum}]^{m} = [yt_{p,n}]^{m}, \quad for \quad m = 1,2,...,M \,,$$

where $[YT^{sum}] = T^{sum}$ from the previous discussion.

Similarly, the Total_Sum for x-slice networks are calculated using the same procedure as done for y-slice networks. The Total_Sum for x-slice networks can be expressed as:

$$[XT^{sum}]^{n} = [xt_{p,m}]^{n}, \quad for \quad n = 1,2,...,N$$

The Total_Sum for x-slice networks can be simply obtained by changing the matrix $A^{m}_{p,n}$ with matrix $B^{n}_{p,m}$ and replace 'n' by 'm' in the above mentioned procedure for y-slice networks. The only difference is that now 'n' is kept constant instead of 'm'.

The Total_Sum ST^{sum} for the small volume $S_{p,m,n}$ is determined as:

$$[ST^{sum}]^{p} = [YT^{sum}]^{m} + [XT^{sum}]^{n}, \quad for \quad p = 1,2,...,P \text{ or}$$

$$[ST^{sum}]^{p} = [st_{m,n}]^{p} = [yt_{p,n}]^{m} + [xt_{p,m}]^{n} \text{ or}$$

$$st_{p,m,n} = yt_{p,m,n} + xt_{p,m,n} \,,$$

where $yt_{p,m,n} = [yt_{p,n}]^{m}$ and $xt_{p,m,n} = [xt_{p,m}]^{n}$

The focus map SF of the small volume $S_{p,m,n}$ is the image frame among the image sequence that gives maximum value of Total_Sum ST^{sum} along the p direction and is expressed as:

$$SF = [sf_{m,n}],$$

$$sf_{m,n} = \arg \max_{p=1,...,P} [st_{m,n}]^{p} = \arg \max_{p=1,...,P} st_{p,m,n}$$

The focus map SF contains the image number for the best focused points of the small volume around the rough estimate. For getting the absolute image number corresponding to the best focused points, the initial rough estimate image number at (x,y) is added to the focus map. Similarly, the focus maps are calculated for all small volumes created by the steps in x and y direction.

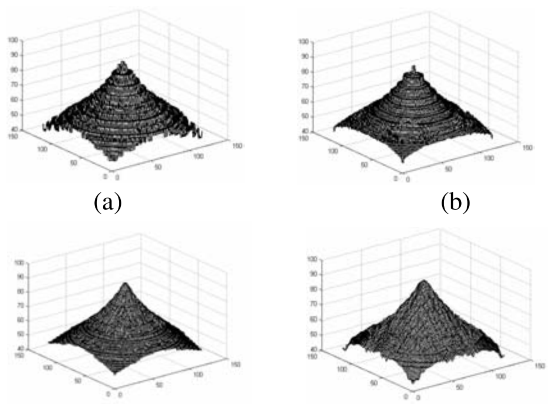
3. SIMULATION RESULTS

Experiments were conducted on a simulated cone, a real cone, and a microscopic object - Lincoln statue on one cent US coin. Figure 2 shows one image from each test sequence. We see in Fig. 2 that only one part of the image is focused, whereas the other parts are blurred to varying degrees.

(a) (b) (c)

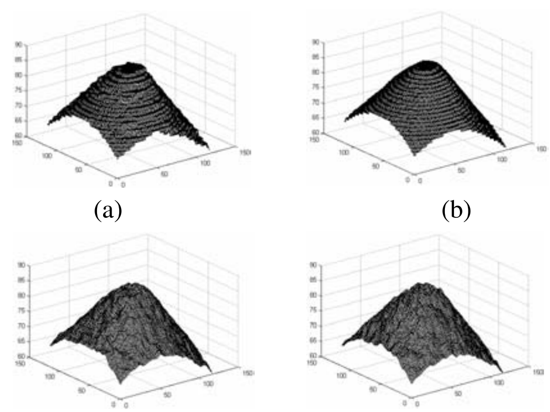
Fig. 2. Images from the test sequences (a) Simulated cone (b) Real Cone (c) Microscopic object.

The 3D shapes or depth maps recovered by SFF methods are shown in Fig. 3-5. The ideal depth map for the simulated cone should be very smooth and the tip should be very sharp. We can see that the depth maps obtained by SFFTR on the simulated and real cones are not smooth. The depth maps seem to change in large jumps instead of varying gradually, and the tips of the cones are not very sharp. The SFFFIS shows better depth maps. SFFDP shows very good results on the simulated cone. The depth map obtained from SFFDP on the simulated cone is very smooth and the tip is very sharp. The new proposed algorithm SFFDP_1 shows much better results as compared to SFFTR and comparable results with those of SFFFIS and SFFDP. The results on the Microscopic object also show that the three algorithms SFFFIS, SFFDP and SFFDP_1 outperform SFFTR. We can subjectively say that SFFDP_1 produces comparable results with those of SFFFIS and SFFDP and better results than that of SFFTR.



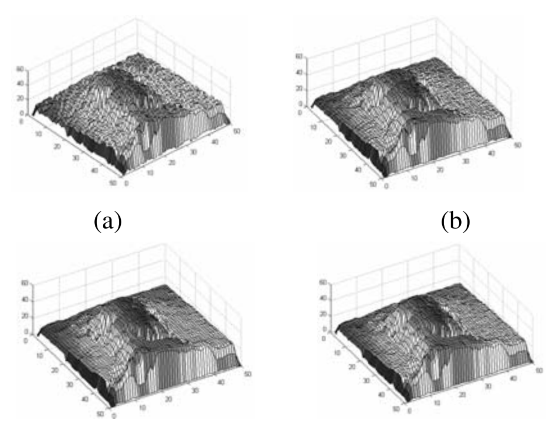
(a) (b) (c) (d)

Fig. 3. The reconstructed 3-D depth map for the Simulated Cone object by (a) SFFTR (b) SFFFIS (c) SFFDP and (d) SFFDP_1.



(a) (b) (c) (d)

Fig. 4. The reconstructed 3-D depth map for the Real cone object by (a) SFFTR (b) SFFFIS (c) SFFDP and (d) SFFDP_1.



(a) (b) (c) (d)

Fig. 5. The reconstructed 3-D depth map for the microscopic object by (a) SFFTR (b) SFFFIS (c) SFFDP and (d) SFFDP_1.

The computer simulation of SFF methods was carried out on 2.8 GHz P-IV PC. The depth estimation time of different algorithms are shown in Table 1 for a sequence of 97 images, and the size of each image frame being 256 x 256 pixels. In the proposed algorithm SFFDP_1, the different parameters selected are: the step size in x and y directions equal to 9, and the number of images around the rough estimate equals to 21. SFFDP used the full image volume for the search of the optimal FIS shape. However, the SFFDP_1 used a small cubic volume around the first estimate for the search of the optimal FIS shape. The number of arithmetic operations reduces considerably in SFFDP_1 as compared to SFFDP. The SFFDP algorithm is much faster than the SFFFIS, but slower than the SFFTR. However, the SFFDP_1, as seen from the Table 1, is so fast that it executes almost in the same time as SFFTR.

Table 1 Comparison of depth estimation time

	SFFTR	SFFFIS	SFFDP	SFFDP_1
Depth estimation Time	4 sec	4 min.	18 sec	5 sec

4. CONCLUSION

The Shape From Focus methods proposed in the literature are either inaccurate or slow. The previously proposed algorithm by authors for 3D shape recovery using Shape From Focus based on dynamic programming optimization technique was more accurate but slower than the traditional method. The new proposed algorithm has shown comparable accurate results with those of previously accurate methods in the literature, and ranks in the fastest algorithms.

5. REFERENCES

[1] E. Krotkov, “Focusing,” International Journal of Computer Vision, Vol. 1 pp. 223-237, 1987.

[2] Shree K. Nayar and Yasuo Nakagawa, "Shape from focus,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 8, August 1994.

[3] Murali Subbarao, and Tae-Sun Choi, “Accurate recovery of three dimensional shape from image focus,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 3, March 1995.

[4] Muhammad Bilal Ahmad and Tae-Sun Choi, “A heuristic approach for finding best focused shape,” IEEE Transactions on Circuit System for Video Technologies (to be published in April 2005 issue).