Converting movie-grade 2D videos to 3D

Abhishek Anand
Department of Computer Science
Cornell University
Ithaca, NY 14850
aa755@cs.cornell.edu

Ashutosh Saxena
Department of Computer Science
Cornell University
Ithaca, NY 14850
asaxena@cs.cornell.edu

Abstract

Over the past few years, 3D display technology has invaded our society in a big way. Even small cities in developing countries now have multiplexes showing movies in 3D format which are seen using special glasses. Apart from the monocular cues that traditional 2D movies provided, this format also provides stereo cues. Human eyes perceive 3D structure of the world mainly by stereo cues which result from disparity between the images of same point formed on left and right retinas. Modern 3D movies like Avatar were shot using stereoscopic cameras(2 cameras side by side). However, we have a wealthy collection of old movies which were not shot using stereoscopic cameras and we have only 2D info for them. In this work, we aim to convert those 2D movies to their pseudo-3D versions by estimating the depths of all pixels in all frames. We combine the monocular, motion and smoothness cues to get a robust depth estimate. So far, we managed to get good depth estimates for videos having some camera motion(atleast for 2 frames in a shot) and nearly static scenes. Even when there is no camera motion(and/or large moving objects), our approach should give decent reasonable depthmaps (thanks to monoclular cues).

1 Related Work

Although not much work has been done on automatically converting 2D movies to 3D, many less-general problems have been addressed in past research. Our work probably the largest amount of motivation from the Make3D work [SSN08] which converts 2D images to 3D. Apart from monocular cues, people have exploited motion cues in a large number of works. Structure from motion(SFM) has been an area of active research for many years. Under various assumptions, people have reconstructed both sparse and dense depths from sets of both ordered and unordered images. The most common assumption is that the scene is static and only camera is moving. Agarwal et al. [ASS$^+$10] reconstruct 3D scenes from a huge collection of 2D pictures assuming (almost)static scenes. Knorr et al.[KKS08] generate 3D stereoscopic videos from 2D videos of static scenes. However, these techniques totally fail when there is no parallax(camera's translation has no component parallel to image plane). By combining monocular and SFM cues, our approach is more robust. Also, we enforce intra and interframe smoothness priors. The ideas is that objects of a scene do not move a lot between adjacent frames of a 30fps video. Finally, we believe that optical flow patterns can be a reasonable predictor of depth and change of depth(Although not implemented yet)

2 Our Approach

The first step is to divide a movie into shots(a set of contiguous frames captured continuously from a camera) A lot of work has been done in the area of shot-boundary detection. We ignore this problem for the time-being and consider only a single shot. Given a shot we model the problem of finding

depth at each pixel and each frame as a regression problem. For this purpose, we use an MRF(a probabilistic undirected graphical model).

Let us assume we have n frames in the shot under consideration.

2.1 Notations

We first describe our notations. We understand that the number of symbols here could be overwhelming. You should read it once now and come back to it when we describe the terms later.

- let P(r,c,k) denote the pixel (r,c) in frame k. Let R(r,c,k) denote the unit vector along the ray joining the centre of procection(COP) to pixel (r,c) in the image plane of frame k. (This ray will also join the COP to the point in 3D world of whose, this pixel is an image)

  Let d(r,c,k) denotes the depth of P(r,c,k).

- Like Make3D, we begin by segmenting the each frame into super-pixels. We approximate each superpixel by a single plane. Every 3D plane can be represented by 3 parameters. Our aim is now to find the (3) parameters for these planes and not find depths for each pixel directly. This gives a huge drop in time/space complexity.

  Let S(r,c,k) denote the superpixel which contains the pixel P(r,c,k). Now we have $P(r, c, k) \in S(r, c, k))$.

- for each frame k, we compute a set N(k) which consists of a pair of pixels $< (r, c), (r', c') >$ which are neighbors(in a 4-connected sense) and belong to different superpixels.

- Let $Conn(S_1, S_2)$ denote the probability that the planes corresponding to the neighboring superpixels S_1 and S_2 of the same frame are connected(unlike cases like those where one plane occludes other). This probability is estimated in the same was as was done in Make3D by considering the similarity of various color,texture features of 2 adjacent superpixels superpixels. (As of now $Conn(S_1, S_2) = 1 \forall S_1, S_2$, but I expect to implement it by 17th)

- Let $\alpha(s)$ denote the plane parameters for the superpixel s. For any point p lying on this plane, $\alpha(s)^T p = 1$. By setting $p = d(r, c, k) * R(r, c, k)$(because p is d(r,c,k) distance away along the unit vector direction R(r,c,k)), we get $d(r, c, k) = 1/R(r, c, k)^T \alpha(S(r, c, k))$.We will be using this equation very very often, So please keep this in mind while reading the rest of this report.

- Make3D gives a depths at each pixel using only monocular cues. We denote these values as $d_M(r, c, k)$

- SIFT[Low99] is a robust way to match points in images which correspond to the same real world point. Given some images, SIFT finds out a small subset of special points which are easy to match because of there unique properties(eg. local maxima of intensity). Let F(k) denote the set pairs of SIFT pixels - first from k^{th} frame and second from $(k + 1)^{th}$ frame such that they correspond to the same physical point.

- Another direct depth cue is obtained from the bundler package. For a subset of those SIFT points, bundler can figure out depths using triangulation from different frames. Let $d_S(r, c, k)$ denote the depth for pixel P(r,c,k) obtained by SFM with a confidence of $C_S(r, c, k)$. If the pixel (r',c',k') is not present in sparse SFM output, we set $C_{sfm}(r', c', k') = 0$. At this point, we still don't know how to compute these confidence values. So $C_S(r, c, k)$ is wither 0 or 1 depending on whether bundler is able to find depth of that point.

We formulate our MRF as:

$$P(\alpha \mid d_M, d_S, C_S, Conn, F, M_f) = T_{make3d} \times T_{sfm} \times T_{planeConnectedness} \times T_{interFrameCoherence} \quad (1)$$

We ignore the normalization function.

2.2 T_{make3d}

This term says that the depth obtained from plane parameters should be similar to Make3d obtained depths.

$$. \quad d(r,c,k) = \frac{1}{R(r,c,k)^T \alpha(S(r,c,k))} \approx d_M(r,c,k) \tag{2}$$

$$\Rightarrow \quad (1 - R(r,c,k)^T \alpha(S(r,c,k)) d_M(r,c,k)) \approx 0 \tag{3}$$

$$\Rightarrow \quad (\frac{1}{\sqrt{d_M(r,c,k)}} - R(r,c,k)^T \alpha(S(r,c,k)) \sqrt{d_M(r,c,k)}) \approx 0 \tag{4}$$

The mystery behind the 3rd step (using square roots) is explained in appendix.

$$T_{make3d} = exp(-\sum_{r,c,k} \{\frac{1}{\sqrt{d_M(r,c,k)}} - (R(r,c,k)^T \alpha(S(r,c,k)))(\sqrt{d_M(r,c,k)})\}^2) \tag{5}$$

2.3 T_{sfm}

Our formulation for depths obtained by bundler(SFM) is almost exactly the same. Only difference is that we multiply the depths by an unknown scale factor s. This is because bundler gives depths at a much different scale than Make3D.

$$T_{sfm} = exp(-w_1 \sum_{r,c,k} C_S(r,c,k) \{\frac{1}{\sqrt{s * d_S(r,c,k)}} - (R(r,c,k)^T \alpha(S(r,c,k)))(\sqrt{s * d_S(r,c,k)})\}^2) \tag{6}$$

2.4 $T_{planeConnectedness}$

This term says that the (finite)planes of adjacent superpixels should be connected at boundaries. As explained section 2.1, (r,c) and (r',c') are 2 neighboring points which are in different superpixels. We want their depths to be same with a confidence Conn(S(r,c,k),S(r',c',k)) which give the probability that the planes of 2 superpixels are connected(and not occluded)

$$T_{planeConnectedness} = exp(-w_2 * X_1)), where \tag{7}$$

$$\begin{aligned} X_1 &= \sum_k \sum_{<(r,c),(r',c')> \in N(k)} Conn(S(r,c,k), S(r',c',k)) \{1/d(r',c',k) - 1/d(r,c,k)\}^2 \\ &= \sum_k \sum_{<(r,c),(r',c')> \in N(k)} Conn(S(r,c,k), S(r',c',k)) \{R(r',c',k)^T \alpha(S(r',c',k)) - R(r,c,k)^T \alpha(S(r,c,k))\}^2 \end{aligned}$$

2.5 $T_{interFrameCoherence}$

SIFT-matching can robustly find some pixels in adjacent frames which correspond to the same physical point in the world. Assuming that things do not change much in adjacent frames(>30fps) of the same shot, these pixels should have same depths

$$T_{interFrameCoherence} = exp(-w_3 * X_2)), where \tag{8}$$

$$\begin{aligned} X_2 &= \sum_{k=1}^{n-1} \sum_{<(r,c),(r',c')> \in F(k)} \{1/d(r',c',k+1) - 1/d(r,c,k)\}^2 \\ &= \sum_{k=1}^{n-1} \sum_{<(r,c),(r',c')> \in F(k)} \{R(r',c',k+1)^T \alpha(S(r',c',k+1)) - R(r,c,k)^T \alpha(S(r,c,k))\}^2 \end{aligned}$$

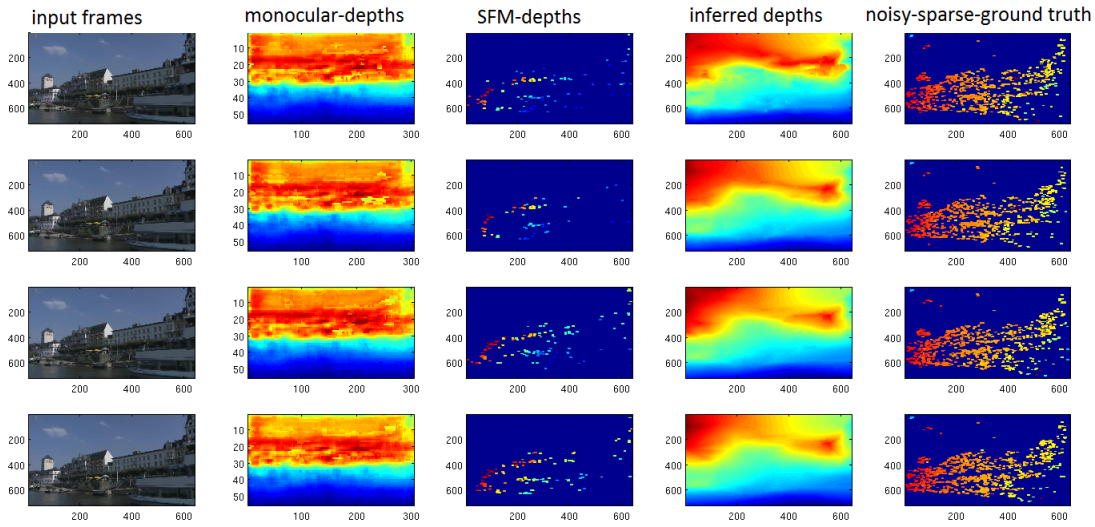

Figure 1: Result on frames of hotel video

2.5.1 Optimization

Clearly, all the terms only involve quadratic terms inside exponent. We ignore the normalization(partition) function. Except T_sfm, all terms can be expressed as:

$$T_p = exp\{-\overrightarrow{\alpha}^T H_p \overrightarrow{\alpha} - \overrightarrow{\alpha}^T b_p - c_p\} \tag{9}$$

where $\overrightarrow{\alpha}$ is one large vector containing all elements of plane parameters for all superpixels for all frames. Recall from section 2.3 that the sfm term also has an additional unknown scaling factor s.

$$T_{sfm} = exp\{-\overrightarrow{\alpha}^T (sH_{sfm}) \overrightarrow{\alpha} - \overrightarrow{\alpha}^T b_{sfm} - c_{sfm}\} \tag{10}$$

H_p and b_p are constant matrices and vectors respectively $\forall p \in \{make3d, sfm, planeConnectedness, interFrameCoherence\}$

$$log(P(\overrightarrow{\alpha}, s \mid d_M, d_S, C_S, Conn, F, M_f)) = -(\overrightarrow{\alpha}^T (\sum_{p \neq sfm} H_p)\overrightarrow{\alpha} + \overrightarrow{\alpha}^T (sH_{sfm})\overrightarrow{\alpha} + \overrightarrow{\alpha}^T (\sum_p b_p) + (\sum_p c_p) \tag{11}$$

Unfortunately this log likelihood is not convex in both $\overrightarrow{\alpha}$ and s. However, if we fix s, it is convex in $\overrightarrow{\alpha}$ - infact, it is a QP. And if we fix $\overrightarrow{\alpha}$, it is convex in s. This suggests an alternating convex minimization algorithm [NH98]. We begin by initializing s =mean of make3d depths/mean of SFM depths. Now, we repeat the following steps till convergence

1. fix s and find optimal $\overrightarrow{\alpha}$. It can be done easily by the matlab quadprog function
2. fix $\overrightarrow{\alpha}$ and find optimal s.

We found that at convergence, s is reasonably independent of initialization value - except when some of the weights(w_1, w_2, w_3) are too high.

3 Quantitive Results

It's difficult to obtain videos with ground truth depthmaps because of many reasons. Most of the depth cameras have a limited range and wont work in outdoor environment. Make3D wont work very good in indoor environments. Other techniques like stereo-matching produce very noise depthmaps. So, we don't yet have a great way to do quantitative evaluation. However, we do have an approximate quantitative evaluation technique. Youtube has a reasonably large collection of videos of

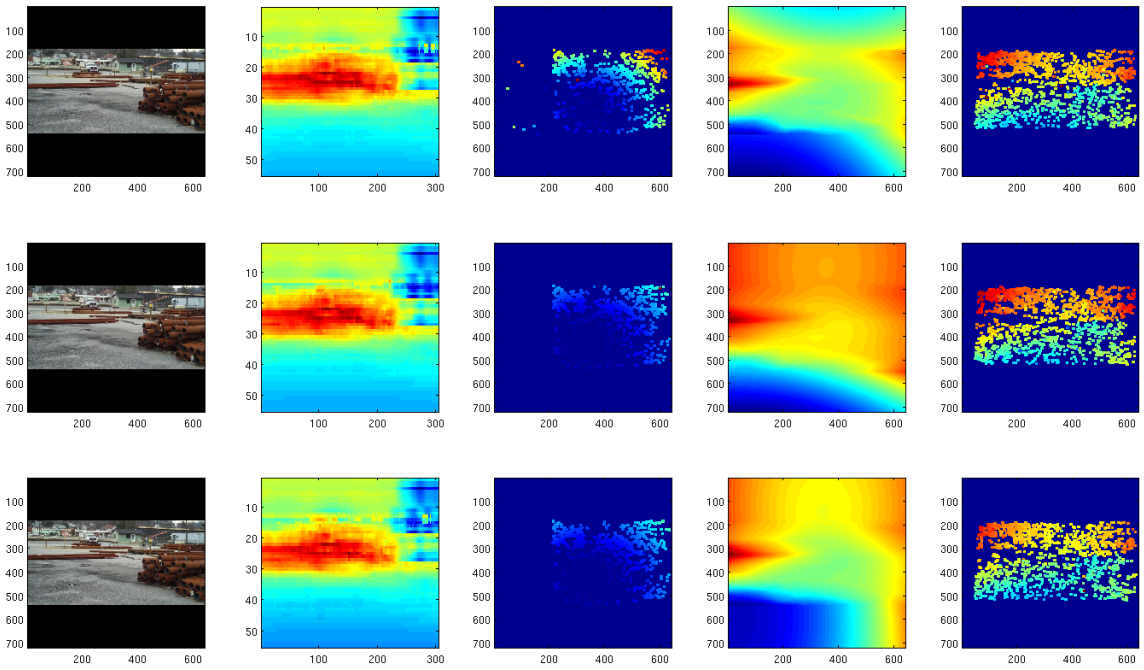

Figure 2: Result on frames of pipes video

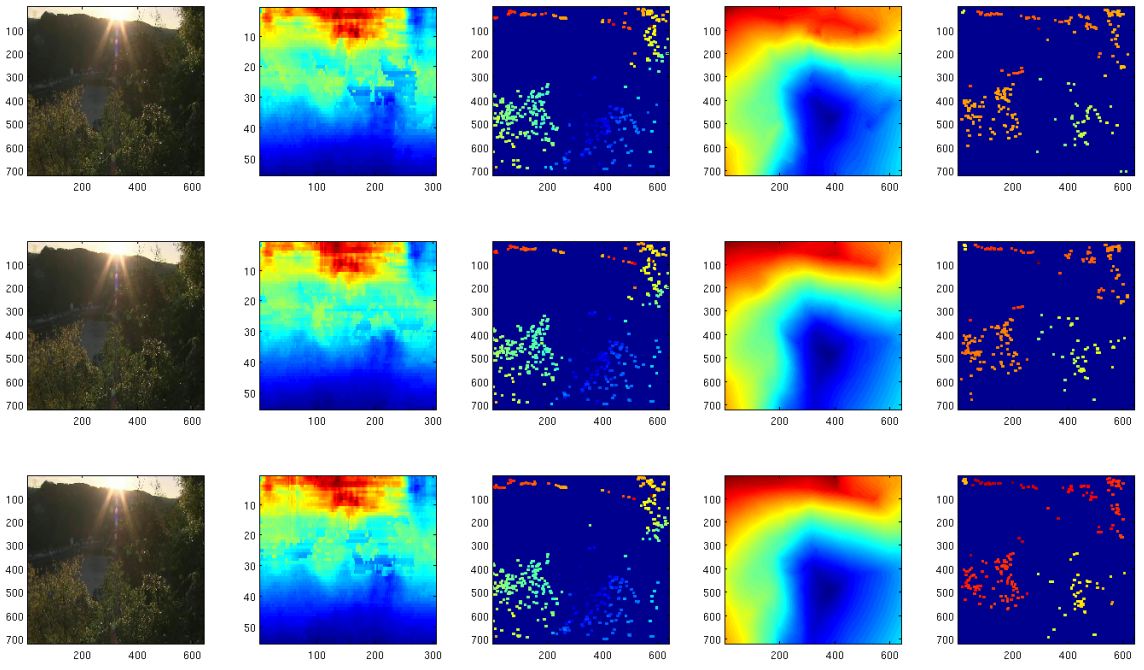

Figure 3: Result on frames of trees video

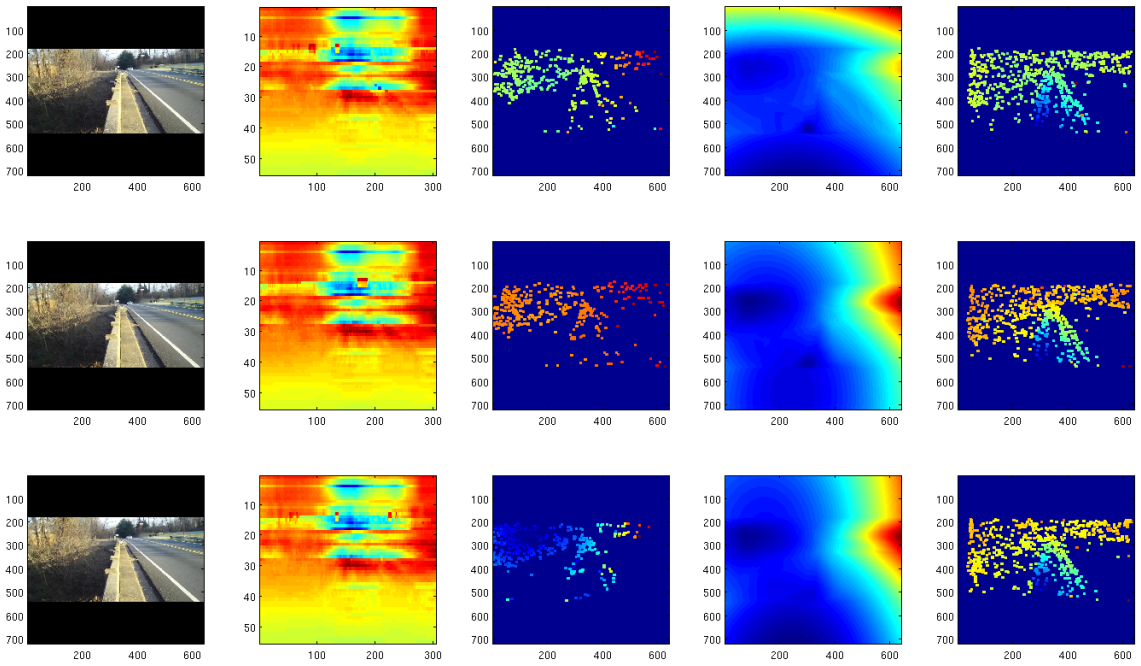

Figure 4: Result on frames of moving car video

video	current implementation	make3dPoint
hotel	62	50
pipes	84	63
trees	79	74
moving car	49	51

Table 1: percentage of pairs ordered correctly

3D format. These can be downloaded in a stereo format(separate video-frames for left and right eyes). We can compute match SIFT points between these 2 frames(left and right). The shift in this point(along with sign) is more for pixels with less depth. So,this gives an ordering of depths of SIFT matches. We consider this as ground truth ordering, although it is noisy. We cannot compute the exact depths because we dont' know the required parameters of cameras like baseline and relative rotation. We evaluate our results by finding the fraction of pairs of points for which their depth ordering is same as ground truth. Based on this metric, we evaluated our approach on 4 videos(attached with submission-email). Table 3 shows the results. In the next section, we show the inferred depthmaps for each case. Note that even an algorithm with predicts depths randomly will get an expected score of 50% on this metric.

4 Qualitative Results

Inferred depths on frames of the 4 videos mentioned above are shown in figures 1,2,3 and 4. The meanings of columns are as mentioned in figure 1. In particular, the depthmaps in last column are obtained by computing disparities by SIFT matching between left and right-eye frames. We will discuss only the hotel video (Fig 1) here. Note that all the depthmaps are in the matlab's default "Jet" colormap in which red corresponds to high values and blue corresponds to low values and green is in between. Note that Make3D output mainly divides the images into 2 parts - lower part has low depths

and upper part has high depths. The 3rd column shows the SFM depthmap output by bundler(using triangulation on all the frames of the shot). Note that this depthmap is sparse. Depths are only available at the small squares. Note that SFM successfully detects that the depths increase when we move horizontally towards LHS. The 4th column shows the output of our approach. Qualitatively, it fuses both the make3d and bundler outputs. Wherever SFM depths are no present(eg bottom right), it take Make3D inputs. At points where both are available it gets something in between. For example, it shows that the depths increase when we move horizontally towards LHS(more red $\Rightarrow$ more depth).

5 Future work

Current results suffer from many problems:

- Bundler gives sparse depthmap. We would try to get a dense depthmap using techniques like PMVS[FP09].
- Current inferred depthmap is over-smoothed. We need to predict occlusions and reduce the weight for those edges while enforcing intra-frame coherency.
- In cases where some object is moving the scene, bundler might give reliable sparse depthmap. This was probably the reason for poor performance on moving car video 4. We need to modify the bundle adjustment process to remove moving objects.
- Motion cues:Probably, depth can be directly predicted from motion cues like optical flow patterns. We need to train a model for this task.
- For proper evaluation, we need to build a repository of videos with ground truth depthmaps - maybe using laser scanners.

References

[ASS+10] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R. Szeliski. Building rome in a day. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 72–79. IEEE, 2010.

[FP09] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 2009.

[KKS08] S. Knorr, M. Kunter, and T. Sikora. Stereoscopic 3D from 2D video with super-resolution capability. *Signal Processing: Image Communication*, 23(9):665–676, 2008.

[Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. Published by the IEEE Computer Society, 1999.

[NH98] R.M. Neal and G.E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89:355–368, 1998.

[SSN08] A. Saxena, M. Sun, and A.Y. Ng. Make3d: learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, pages 824–840, 2008.