Fast View Interpolation from Stereo: Simpler can be Better

N.Martin and S. Roy
{martinic,roys}@iro.umontreal.ca
DIRO Université de Montréal

Abstract

In this paper, we propose to rely only on images to generate novel views, and recall why modeling of the complete scene is often too expensive in the context of view interpolation of simple scenes. We investigate ways to achieve view interpolation by mean of forward and backward mapping of disparity maps. We present situations in which each one requires less computations and gives better results. Contrary to what we might expect, very simple stereo algorithms can produce very convincing interpolation despite providing really bad disparity maps. We propose to explain this with a probabilistic model of depth discontinuities. We test this model on synthetic data created to fit real image statistics and compare with images widely used in stereo[13]. In practice, forward and backward mapping methods can rely on simple stereo algorithms running in real time, to produce very good results. A sequence of real images was acquired to allow accurate comparison of interpolated images, and standard metrics are used to assess the quality.

1. Introduction

The process of synthesizing novel images of a scene from a different point of view than the ones given as references, is referred to as view interpolation. There exists a lot of different techniques to achieve such a goal, ranging from 3D model based rendering to image based rendering. This work focus on using at least two images of a scene as the reference views, using standard stereo algorithms to retrieve depth informations, and generate novel views. We won't explore all computer graphics based methods, and notably we won't compare to methods where full modeling of the scene is required. Many authors already suggested that this is not generally the best approach, first because model rendering is bounded by scene complexity whereas image interpolation is bounded by image resolution [2], and storage of 3D models can sometimes be too large in the case of very complex scenes. Secondly it tends to give worst results because automatic modeling of scene requires concessions to be made in order to make the model coherent with all views, but when rendered from the point of view of a virtual camera, it is not necessarily consistent anymore. We also won't compare to methods using complex functions describing parts of the scene visible from a point of view (in particular all methods based on plenoptic functions [9, 15, 5]), because our work supposes simple scenes for which much simpler methods can be applied.

Stereo algorithms have to deal with a lot of problems to accurately recover the disparity information at each pixel. Most of those problems are caused by the presence of occlusions in the scene, i.e. pixels that correspond to objects visible in only some of the reference views. In the standard two cameras stereo setting, depth information at disparity discontinuities can not be recovered exactly, unless disparity maps are labeled by hand as in [6]. When more camera views are available, a visibility scheme can be set to select only cameras that can see the object[10]. One of the goal of this work is to show that even if disparity maps obtained by the algorithm are far from perfect, the image interpolated is still of great quality. Thus we will use only the simplest stereo algorithm, that do not handle visibility and occlusion checks. It is important to recall that our work focuses on view interpolation in the case of very simple scenes, that is where reference views are close from each others, and the angle of view spanned by all reference views is really small. Other problems stereo algorithms face are texture ambiguity due to image sampling, or repeated patterns of intensity. In fact, most of those problems can be solved by incorporating smoothing between pixels in the cost function.

Most of the applications of view interpolation are in real-time contexts. This rules out graph cuts or believe propagation algorithms even though they seem to perform best according to recent reviews by Scharstein and Szeliski[12], as they are very hard to implement in real time, and even when efficiently implemented they dot not handle well very searching over a big disparity range, necessary for teleconferencing applications where objects are most of the time really close to the references cameras[3]. However , several other stereo algorithms can be implemented in real-time, such as direct search and dynamic programming [17]. We

implemented both of them in this paper for our experiments.

2. Previous work in image interpolation

Chen and Williams[2] presented an approach to produce interpolated novel views by interpolation of images using correspondences given by range data for each camera. They create two maps containing per pixels correspondences for each pairs of cameras, which are represented as 3D offset vectors, that are linearly interpolated to produce intermediate views. We refer to this kind of interpolation algorithm as forward mapping algorithms, and the authors already mentioned that overlapping and holes created by this method are their major drawbacks. Most of the authors handle holes by filling them with colors interpolated from the neighborhood. Scharstein [11] explained that using both disparity maps decreases the number of holes. When disparity maps are perfect, only holes corresponding to fully occluded pixels will remain. Overlapping of pixels are most of the time eliminated by use of z-buffering or visibility handling.

Backward mapping methods were introduced in the context of image warping by Wolberg [18]. Laveau and Faugeras [7] proposed a framework in which view interpolation is possible without explicit 3D reconstruction using backward mapping. Many other authors have however tried to generate a model of objects to be reconstructed prior to rendering it from the point of view of the interpolated camera [6], but as we have already mentioned, we intend not to depend on a prior full 3D reconstruction of the scene.

Stereo vision is not always used to obtain correspondences prior to image interpolation [2, 7, 14]. However, since it is the main theme of this article, we will focus on methods using stereo as part of the interpolation process. Scharstein [11] mentioned that it is important to adapt a stereo algorithm for the purpose of view synthesis. It has already been widely said that for view interpolation, it is more important to obtain good color for the interpolated view than good disparities [21, 3, 11], and that's what the stereo algorithm should focus on. We will try to show that simple algorithms tend to do this naturally because they are not designed to get the best disparity map, but just find good correspondences. This is important, because it explains why we will only experiment with the two basic stereo algorithms direct search (DS) and dynamic programming (DP). Other authors already explained why very sophisticated algorithm are not well suited for image interpolation, and especially in real-time context [3]. An important part of this article is the applicability of methods for real-time view interpolation, and many authors presented ways of doing real-time stereo, most of the time on commodity graphics hardware [19, 17].

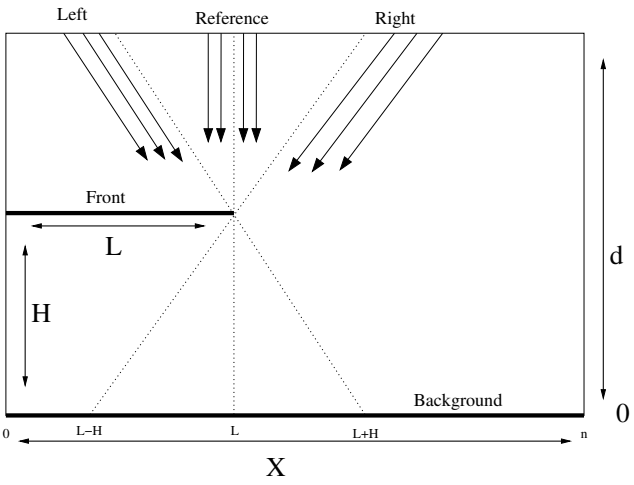

Figure 1. Discontinuity model. An object *Front* is at height H above *Background*. We intend to estimate the best disparity along each ray of the *Reference* image, by projecting onto the *Left* and *Right* images.

3. A probabilistic model of matching near depth discontinuity

Since depth discontinuities play a major role in image interpolation, it is important to understand better the behavior of stereo cost functions near discontinuities. We propose a simple discontinuity model, illustrated in Figure 1, and an image formation model with image properties similar to those of real images, to help perform simulations. It is important to note that a lot of work has already been done on probabilistic models for stereo and IBR [1, 20], however it was in the context of using it in an algorithm to improve its performance. We propose a model to justify why simple stereo algorithm can produce good results.

The discontinuity model assumes that a flat front object is observed floating above a background. For a reference view pixel x, we must compute its best depth $d(x)$ using two images (left I_L and right I_R). The cost function for assigning depth d to pixel x is

$$c(x, d) = |I_L(x + d(x)) - I_R(x - d(x))|.$$

The images are obtained by observing the front and background objects, which have colors values $I_F(x)$ and $I_B(x)$ derived from their own distributions. These intensity distribution follows a Brownian motion equation in the space of intensities, with time representing the distance in pixels from a seed point. The mean intensity $\bar{I}$ and diffusion coefficient σ are specific to the front or background objects. The reason for using brownian motion is that it best describes

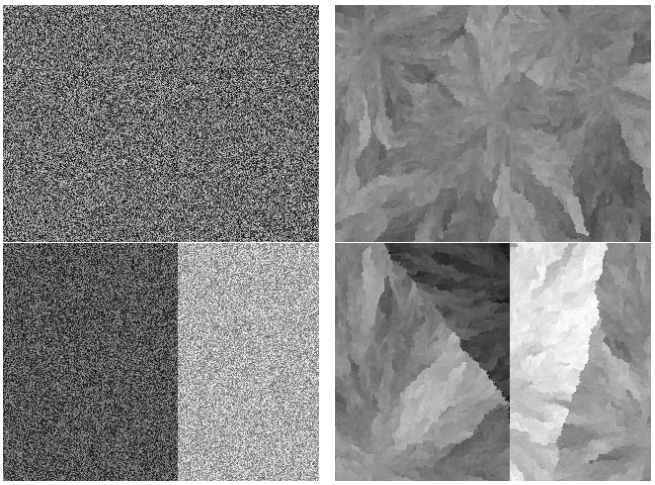

Figure 2. Synthetic image formation model. Each image is the left image of a simulated stereo pair. The two top images feature the same average intensity value $\bar{I}$. The two left images have independent pixel intensities ($\sigma = \infty$) while the two right images have $\sigma = 3$.

how the difference of image intensities on a single object depends on the distance between them. This is observed in almost all natural scenes (see bottom of Figure 3). In our model, two objects (front and back) are each described by two brownian parameters $(\bar{I}, \sigma)$, the mean intensity $\bar{I}$ and the diffusion coefficient σ, to yield two images $I_F()$ and $I_B()$ for the front and background respectively.

Notice that if σ is 0, then an image with flat intensity $\bar{I}$ is obtained; if σ is very high, then pure noise is obtained. It is thus possible to model all levels of "local texture randomness" with this simple model. Examples of generated images are provided in Figure 2.

In the discontinuity model of Figure 1, the front object has length L, at height H, so we obtain the three image functions

$$I_M(x) = \begin{cases} I_F(x) & (x \leq L) \\ I_B(x) & \text{otherwise} \end{cases}$$

$$I_L(x) = \begin{cases} I_F(x-H) & (x \leq L+H) \\ I_B(x) & \text{otherwise} \end{cases}$$

$$I_R(x) = \begin{cases} I_F(x+H) & (x \leq L-H) \\ I_B(x) & \text{otherwise} \end{cases}$$

where only I_L and I_R are available and I_M is the image that is to be interpolated. Notice that if $H = 0$, the front object is on the background and all images are the same.

In order to verify if the brownian image formation model is accurate, we propose to verify that synthetic and real images follow these properties:

- The intensity similarity between pixels that belong to different objects is independent.
- The intensity similarity between pixels that belong to the same object is highly dependent on the pixel separation.

We measured these properties in the four synthetic images of Figure 2, as well as for two real images (teddy and tsukuba). We computed, for varying pixel distances r between 1 and 50 the probability density of intensity differences between pixels. We expect that the distributions will become wider with increased distance r. Results are illustrated in Figure 3. We observe that the real images distributions (at bottom) match closely the ones of the top two right images, where the brownian intensity model was used.

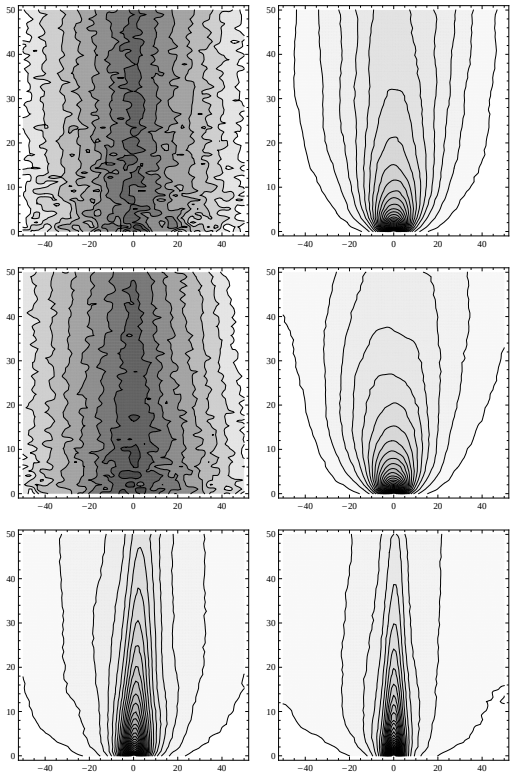

Figure 3. Distributions of local pixel intensity differences (horizontal axis), for varying distance between pixels (vertical axis). The top four corresponds to the images of Figure 2, while the bottom two are for real images `teddy` and `tsukuba`

The cost function of our discontinuity model were computed for the four synthetic images of Figure 2, and are dis-

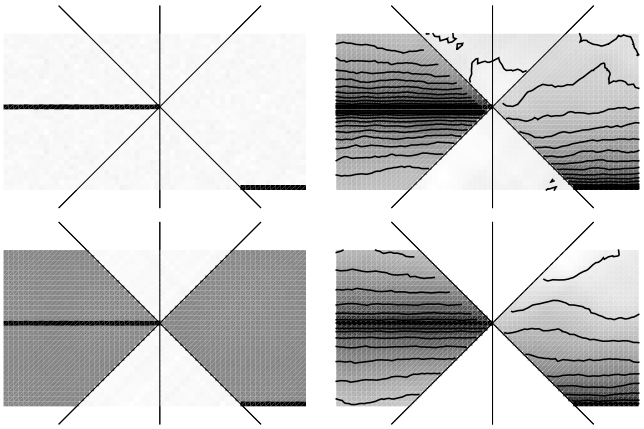

Figure 4. Simulation of cost functions near a discontinuity. The cost functions (horizontal axes is x, vertical is disparity d) corresponding to the discontinuity model. The results corresponds to the images of Figure 2. The lowest cost is black, the highest cost is white. The absence of iso curve means that the cost function is flat.

played in Figure 4. The white triangle is the area where the cost is quite high and where no algorithm would ever pick a disparity solution. For the two results on the left, the images had no "local pixel correlation" and thus featured flat cost functions. Any disparity can be chosen in those area. For the two results on the right, the cost functions show a gradient that favors picking the smallest disparity possible along the line of occlusion induced by the front object.

Based on these observations, we propose that the local pixel correlation observed in real images makes simple stereo algorithm behave in a more predictable way than expected and this might explain why they do not break down at discontinuities but rather go down a slope that links the closest object to the farthest one. Even if these disparities are wrong, they are quite adequate for view interpolation.

4. Solving view interpolation using stereo vision

For the sake of simplicity of this section, we will assume that the two image are rectified, i.e the epipolar lines are y-aligned between the two images. However it can easily be extended to cases where images are not rectified by applying a simple rectification step to the two images [4, 11]. We also present algorithms to be applied for each scanlines so it is independent of the y values of a pixel.

From two images I_L and I_R, image interpolation aims at generating images corresponding to viewpoints lying on the line joining the two views. There are two different approaches to generate I_M, the image of the virtual view that is to be interpolated. Most mapping problems (rotation, warping [18]) can be solved using either forward or backward approach, however most of the time backward mapping is better suited as pointed out by Lei and Hendriks [8]. In the following subsections, D_L and D_R refer to disparity map for the left and right view and D_M the one of the interpolated view. α is a coefficient that represents the distance of the virtual camera on the straight line between left and right cameras. If $\alpha = 0$, then the virtual camera is the left camera, and if $\alpha = 1$, it is the right one.

4.1 Forward mapping

Scharstein [11] described in his thesis, a simple algorithm to produce an image representing the view of a virtual camera using dense stereo matching. Once disparity maps for left D_L and right D_R views have been computed, he proposes to use each of them to vote disparities in a third map D_M. The algorithm to vote using only the left disparity map uses the following formula at each pixels of the interpolated image to compute image intensity :

$$I_M(x - \alpha D_L(x)) \quad = \quad I_L(x)$$

Figure 5. Glitches due to aliasing when no real visibility handling is done during votes, keep smallest disparities (Left) or largest disparities (Right)

This process produces disparity maps where disparity can be voted at a position that lies between two pixels. Furthermore, we already know that such maps contains holes. By using both maps, part of the holes can be filled out (holes corresponding to partially occluded pixels are filled out [11]), but a process of visibility has to be carried out, because several different disparities can be voted at the same place (also referred to as overlapping [2]). If both disparity maps are used then the following algorithm can be used :

for all x **do**
  $vis(x) \leftarrow visibility(D_L, D_R)$
  if $vis = left$ **then**
    $I_M(x - \alpha D_L(x)) = I_L(x))$
  else if $vis = right$ **then**
    $I_M(x + (1-\alpha)D_R(x)) = I_R(x))$
  else
    $I_M(x+(1-\alpha)D_R(x)) = \alpha\ I_L(x)+(1-\alpha)\ I_R(x)$
  end if
end for

The visibility function used here is to be chosen to express which camera can be used to choose the right color. Fig. 5 shows that simple rules like choosing the camera that vote for the smallest or the largest disparities produce haloing (shadow parts of objects are visible because of the discrete nature of voting scheme) or smearing (foreground and background objects are merged when disparity is wrong) [11], and serious visibility handling has to be made to get correct results. Some implementations use the direction of the camera movement to handle such visibility problem by voting only in a certain order, thus discarding successive votes in the same pixels. Finally, after both algorithms (using one or both disparity maps), remaining holes have to be filled in some ways.

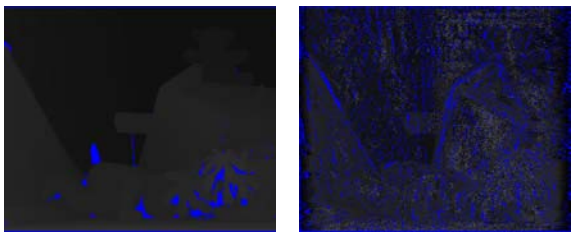

Figure 6. Disparity maps for the interpolated view obtained by voting with both groundthruth images(Left), or two computed disparity maps (Right)

The biggest problem about this method is that it requires both disparity map to be coherent, that is some sort of round-trip has to be respected. If the disparity maps used to vote are really not accurate, gaps are created more often, and at random position, which create more and more holes to be filled as shown by Fig. 6, where two disparity maps are compared, the one obtained by forward mapping of both groundtruth from the teddy sequence[13], and the one obtained by forward mapping of disparity maps computed with direct-search. The main advantage of such a method, is that once the two disparity maps corresponding to reference views have been calculated, new views can be generated easily by the previous voting scheme.

4.2 Backward mapping

Backward mapping projects back disparities obtained from the points of view of a virtual camera located between two real cameras. The following formula might be use to do so :

$$I_M(x) \quad = \quad \alpha I_L(x + D_M(x)) + (1-\alpha) I_R(x - D_M(x))$$

Notice that for each pixel of the virtual view $I_M(x)$, it is deprojected in both camera images (I_L and I_R), and the final intensity value is a linear blending of the intensities from those images. This may seem to give bad results, because when objects are occluded, the pixels representing them are expected to be different in both images, since they are not fully visible. However, this is not the case in practice. The disparity obtained for each occluded pixel makes this pixel effectively visible and as such, assigns it a color that is coherent between the images, even if it is the wrong color. Given local texture coherence, this color is probably close to the real one. The main problem is now to find this disparity map, and the standard algorithm does so using the following equation to compute a disparity at each pixel of the virtual view :

$$D_M(x) \quad = \quad \arg\min_{d \in R} f(I_L(x - \alpha d), I_R(x + (1-\alpha)d))$$

in which R is the range of valid disparities to search for and f is a function of match between two intensity values.

The main disadvantage of this approach is that it requires the computation of a complete disparity map for each view to be interpolated, which is more computationally intensive. Nonetheless, since the search for the best disparity is done for each pixel of the destination virtual view, it produces dense disparity map, with no holes to fill. No management of overlapping is required as well, since only one disparity is chosen at each pixel. That's two of the reason why backward mapping is most of the time best suited for view interpolation.

4.3 Comparing forward and backward interpolation

In previous sections we presented forward and backward methods, each with its strengths and weaknesses. Another way to compare them is with regard to the context of the interpolation. For example, we mentionned that forward mapping requires the calculation of disparity maps for both (or more) reference views only once, and then any view can be interpolated almost instantaneously by a simple voting scheme. However, in a context where reference views are always changing, like teleconferencing or any real-time view interpolation from live camera feeds, forward mapping still requires those maps to be computed, but this time once

every frame! So in this context, it becomes more costly to use forward mapping, because backward mapping requires a single disparity map computation from the point of view the interpolated camera.

In remote view interpolation appliquations, the reference images are sent to a distant client over a network, where it is then interpolated locally by the client. This is typical of web based virtual navigation of real environment. In this case, forward mapping requires that additional disparity data is sent together with the image data, thus increasing the bandwidth required. On the other end, backward mapping can accomodate the image stream without additional information and relieves the image provider from running any stereo at all.

5. Experiments

In this section, we intend to compare both forward (FM) and backward mapping (BM) for both simple stereo algorithm we presented (DS and DP). Determining which interpolation method yields better results is not as easy as directly comparing the interpolation image computed by those methods. Szeliski [16] said that a perceptually-based metric should be used to account for the quality of an interpolated image. It has also been reported that using a metric that estimates the quality of a whole interpolated sequence instead of a single image would be ideal [11]. However, no real metric has already been proposed to do so, and that's why we used RMS and T, defined in [12] as :

$$RMS = \left(\frac{1}{N} \sum_{(x,y)} (I(x,y) - \hat{I}(x,y))^2 \right)^{1/2}$$

$$T = \frac{1}{N} \sum_{(x,y)} (|I(x,y) - \hat{I}(x,y)| > \delta)$$

where $I(x,y)$ is the color intensity of a pixel in the real image, $\hat{I}(x,y)$ is the color intensity of a pixel in the interpolated image, N is the total number of pixels, and δ is a threshold (equal to 15 in our experiments). These measures will compare our interpolated frames with the groundtruth.

We made a sequence of a scene filmed by a camera translated on a rail to produce perfectly aligned data. We took a frame at every millimeter of camera displacement, in order to provide groundtruth frames to compare with. Notice that Szeliski and Scharstein [12] proposed to assess the quality of a depth map generated by a stereo algorithm by generating novel views corresponding to groundtruth data, and to estimate the relative error. We did not do this because we already mentioned that for view interpolation, disparity errors do not necessary create big errors in the generated novel views, and there are in fact a lot of different disparity maps that can generate the exact same groundtruth frame (especially in textureless areas). As a matter of fact, we do not think that comparing the disparity map generated for the novel view to the groundtruth disparity map (if present) is relevent, as it is not necessarily a good measure of how good the interpolation will be.

Direct search (DS) and dynamic programming (DP) have already been implemented for real time performance by previous authors [19, 17] but not necessarly in the context of view interpolation. For images of size 320×240 with 32 disparity levels, we acheive 25 interpolated images per second on backward mapping with dynamic programming, while direct search acheives 85 interpolated images per second. Forward mapping run as fast as the user wants it to. The hardware used was a NVIDIA GeForce 7400 graphics card, installed on a laptop featuring an Intel CoreDuo 1.83Ghz.

We made two experiments. The first one features two images separated by a baseline of 30mm, as illustrated in Fig. 7, and interpolated a frame at each millimeters, for which groundtruth is available. For the second experiment, the *baseline test*, a single frame is interpolated for a succession of larger and larger baselines, while remaining at the exact middle between the two reference views. The groundtruth is available for that frame.

Figure 7. Two images from the labo sequence used for our tests.

5.1 Interpolation experiment

Fig. 8 presents the results we obtained during the first experiment, comparing the RMS measure for combination of both DS and DP with FM and BM. For comparison, we also include for this experience the results of the previous metrics for the dissolve algorithm (D) which is simply a fade between both images (same as interpolating with a disparity map where all objects are at infinity). It corresponds in our opinion to the worst case interpolation algorithm, so it defines a limit to the error any interpolation method should do. D performs worst than every method, which is hopefully good news. We measured the performance on other sequences, and obtained the results of Fig. 9, where the naive D algorithm was removed to help differenciate the four other algorithms. Also, the T-values were removed since they yield about the same results than RMS.

In both results, it is clear that BM-DP performs best for every sequence, regardless of the metric used. FM-DP gives also most of the time good results (second best in all cases). Direct search (DS) gives about the same results with backward (BM) or forward (FM) mapping, even though we would have thought that BM would always be better than FM. This can easily be explained by the fact that the DS algorithm is easily corrupted by local ambiguity in repetitive texture which creates noisy results. These noisy results give about the same poor metric values, regardless of the mapping method used.

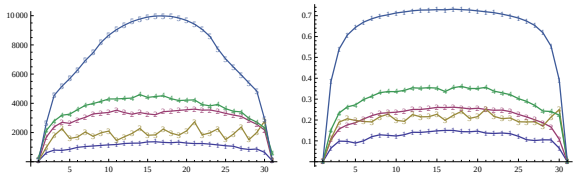

Figure 8. (Left) From bottom to top : RMS values for BM-DP, FM-DP, BM-DS, FM-DS, D. (Right) From bottom to top : T values for BM-DP, FM-DP, BM-DS, FM-DS, D.

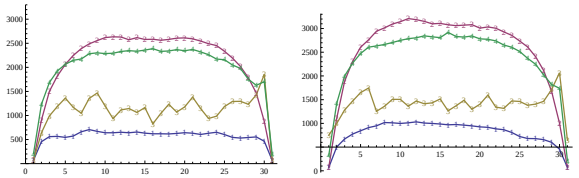

Figure 9. From bottom to top : RMS values for BM-DP, FM-DP, FM-DS, BM-DS in frames 260-290 (Left) and 340-370 (Right)

5.2 Baseline experiment

For the second experiment, we picked a single frame and interpolated it from two reference views equally separated around it, for increasing baseline values. In our opinion, increasing the baseline test well the robustness of an interpolation algorithm and provide a direct measure of it usefullness. An algorithm that performs well at large baselines requires less images of an environement to provide the same interpolated experience. In practice, we picked a frame at 100mm which was then interpolated from reference views at $100 \pm k$ mm, with $1 \leq k \leq 30$. At 30mm, the disparity range is becoming too large to be solved in real-time. We also tested frames at 200mm and 300mm.

The results are presented in Fig. 10, where we plotted the RMS values for each combination of algorithm and mapping method. Those results confirm the previous one, in the sense that DP-BM always gives better results whatever the baseline and the order is approximately the same for the rest of the methods. However, we thought that DP would be much less sensitive to baseline variation than DS since it enforces smoothness. This is not the case as we see the curves all increasing at about the same rate.

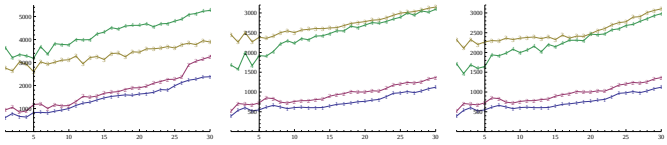

Figure 10. From bottom to top : RMS values for BM-DP, FM-DP, BM-DS, FM-DS for interpolation with frame 100 (Left), 200 (Middle), 300 (Right) chosen as virtual view, and for growing values of baselines.

In order to test the stability of the interpolated image as we increase the baseline, we compared interpolated images for successive baseline values rather than with the groundtruth. Fig. 11 is a plot of errors between successive frames, showing that FM is less stable than BM. However, the situation for FM is even worse. In fact, not only the error between successive frames is higher, but it is distributed mostly around the edge (bottom left of Fig. 11) while BM distributes the error almost uniformly (bottom right of Fig. 11). This does not show up in the RMS values, but it is very perceptible visually. This confirms the need for a better metric for perceptual comparison of interpolation results.

Figure 11. (left) Plot of RMS difference between interpolated frame with successive baseline, (middle) image difference between two interpolated frame for successive baselines with FM, (right) and BM (black means more error).

6. Conclusion

We compared combination of both mapping methods and stereo algorithms to produce convincing view interpolation. The results show that the backward mapping always give better results than the forward mapping. This may of course be because we did not implement serious visibility handling (part of this work was to show that backward mapping can generate very good results without occlusion nor visibility handling), but then again, it would have required better stereo algorithms to generate more accurate disparity maps to vote from. It has been made clear that in some cases, forward mapping is really convenient because of its computational simplicity, but it leaves difficult problems to solve. We presented a probabilistic model that can explain why even very naive stereo algorithms can gives good interpolation. It does not however prevent stereo algorithm to be fooled by texture ambiguity and such, and that's why more elaborate stereo method may perform better. We generated interpolated images corresponding to virtual cameras for which we have the groundtruth images. Metrics used for our experiments do not necessary agree with what the perceptual judgement of a human might have. Perceptually based metric are still required to assess quality of sequences of interpolated images. Finally we hope in the future to further improve the results obtained with dynamic programming which from our point of view is the most simple yet effective algorithm to use for view interpolation. We think that in context of static content to be interpolated, we could preprocess datas to incorporate informations that could help the dynamic programming process to produce even better interpolation, still in real time.

References

[1] P. Belhumeur. A bayesian-approach to binocular stereopsis. *Int. J. Computer Vision*, 19(3):237–260, August 1996.

[2] S. E. Chen and L. Williams. View interpolation for image synthesis. *Computer Graphics*, 27(Annual Conference Series):279–288, 1993.

[3] A. Criminisi, J. Shotton, A. Blake, and P. Torr. Gaze manipulation for oneto -one teleconferencing, 2003.

[4] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Mach. Vision Appl.*, 12(1):16–22, 2000.

[5] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *SIGGraph-96*, pages 43–54, 1996.

[6] T. Kanade, P. Rander, S. Vedula, and H. Saito. Virtualized reality: Digitizing a 3d time-varying event as is and in real time. In H. T. Yuichi Ohta, editor, *Mixed Reality, Merging Real and Virtual Worlds*, pages 41–57. Springer-Verlag, 1999.

[7] S. Laveau and O. Faugeras. 3-D scene representation as a collection of images and fundamental matrices. Technical Report RR-2205.

[8] B. J. Lei and E. A. Hendriks. Real-time multi-step view reconstruction for a virtual teleconference system. *EURASIP J. Appl. Signal Process.*, 2002(1):1067–1087, 2002.

[9] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 39–46, New York, NY, USA, 1995. ACM.

[10] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo – occlusion patterns in camera matrix. In *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 371, Washington, DC, USA, 1996. IEEE Computer Society.

[11] D. Scharstein. *View synthesis using stereo vision*. PhD thesis, Ithaca, NY, USA, 1997.

[12] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[13] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. pages I: 195–202, 2003.

[14] S. M. Seitz and C. R. Dyer. Physically-Valid View Synthesis by Image Interpolation. In *Proc. Workshop on Representation of Visual Scenes*. IEEE Computer Society Press, June 1995.

[15] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 299–306, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[16] R. Szeliski. Prediction error as a quality metric for motion and stereo. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 781, Washington, DC, USA, 1999. IEEE Computer Society.

[17] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 798–805, Washington, DC, USA, 2006. IEEE Computer Society.

[18] G. Wolberg. *Digital Image Warping*. IEEE Press, July 1990.

[19] R. Yang and M. Pollefeys. Multi-resolution real-time stereo on commodity graphics hardware, 2003.

[20] A. Zisserman, Y. Wexler, and A. Fitzgibbon. Image-based rendering using image-based priors. pages 1176–1183, 2003.

[21] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 600–608, New York, NY, USA, 2004. ACM.