SIMULATION AND REAL-TIME VISUALIZATION OF CHANGING BASELINE IN A STEREO PAIR

Matías Rodríguez and Javier Preciozzi
InCo, Fac. de Ingeniería, UdelaR, Uruguay
{rodmati,jprecio}@adinet.com.uy

Gabriele Facciolo
D.Tecn., UPF, Barcelona, Spain
gabriele.facciolo@upf.edu

Andrés Almansa
TELECOM ParisTech, France
andres.almansa@enst.fr

ABSTRACT
The benefits of photogrammetry from low baseline stereo pairs have been increasingly demonstrated in recent years by engineers at the French Space Agency (CNES) and their collaborators, thanks to new advances in image restoration and computer vision. Such an emerging technology requires new tools for generating test data, evaluating and visualizing results. This article discusses the mathematical, numerical and computational problems involved in simulating such low baseline stereo pairs with sufficient accuracy, as well as in the inverse scenario, where the disparity map computed from a low-baseline stereo pair has to be expanded to generate a large baseline stereo pair that makes visual evaluation of the result by photogrametric experts easier. The trade-offs that have to be made in order to obtain accurate results on very large images, with tight constraints on computer resources are also discussed.

KEY WORDS
image processing, 3D visualization, stereo, simulation

1 Introduction

Air-borne or satellite earth observation systems usually take so-called “stereo-pairs” of images of the same part of the ground with the aim of finding 2D point correspondences between both images, and then deducing the 3D coordinates of such points by triangulation. If the image pair is rectified to epipolar geometry [5], and if the altitude H of the camera is much bigger than the altitude h of an object in the observed scene, then its height h can be deduced from the disparity δ between the projected locations of the object in both images u and $\tilde{u}$ via the following expression [3, 5]:

$$\delta[\text{pixels}] = \frac{B}{H}\frac{1}{R}h[\text{meters}]$$

where B is the baseline (distance between both viewpoints) and R is the size of a pixel projected on the ground.

The classical approach adopted by most algorithms for photogrammetric digital elevation model (DEM) computation, is to consider that disparities can only be estimated up to pixel precision. Since the resolution R is limited by hardware constraints, this assumption leads to the use of relatively large B/H ratios (about 1) as the only way to improve the accuracy in height. This approach presents several disadvantages, especially in urban areas; Therefore, engineers at the French Space Agency (CNES) and their scientific collaborators have recently demonstrated the interest and feasibility of small baseline stereo in urban environments [9, 2, 1, 6, 3].

Motivation. In order to quantitatively evaluate automated small baseline stereo reconstruction algorithms, we need to compare the estimated disparities with some reliable ground truth. But due to its emergent technology status, high quality low-baseline stereo pairs with a correspondingly accurately registered ground truth do not exist and are quite difficult to build in a consistent manner. For this reason, there is an urgent need for simulated small-baseline stereo pairs and their corresponding ground-truth disparities. These can be built either from a more standard large-baseline stereo-pair, or from an orthophoto and the corresponding DEM. This paper discusses precisely how such simulations can be modelled and implemented in practice. As a second application scenario, interactive large from small baseline simulations can be used to present these new results to photogrammetric experts that are both unused and highly reticent to the small baseline setting. This way, the message that small-baseline stereo actually works shall be conveyed to a larger public.

Requirements. The simulation obtained in the first case will be used as the input for sub-pixel algorithms (small baseline approach). Thus, the results should be artifact-free, *but also* highly accurate.
In the second case the output should be suitable for exploration by the human eye, so an artifact-free simulation is still important but geometric and radiometric accuracy constraints are less significant. On the other hand this application scenario introduces real-time (interactivity) constraints, and the requirement to allow for interactive navigation (through pan, zoom and altitude change) of very large datasets. This imposes very tight computational constraints (both in terms of CPU time, memory usage, and I/O operations) far more demanding than similar visualization applications (like 3D games), where the 3D model can be drastically simplified in advance by an artist.

2 The effect of baseline on stereo pairs

Let us give here an overview of the different aspects that have to be taken into account when modelling viewpoint changes or baseline changes. Numerical simulation details will be left for the following three Sections.

Geometric distortion The advantage of using stereo pairs with low B/H values can be formulated more precisely in the following manner. In the ideal case where $B/H \to 0$, no occlusions occur. Given an image pair $v, \tilde{v} : \mathbb{R}^2 \to \mathbb{R}$ under these conditions the relation between them is captured by a simple image deformation model

$$v(\mathbf{x}) = g(\tilde{v}(\underbrace{\mathbf{x} + \delta(\mathbf{x})}_{\Phi(\mathbf{x})})) = g \circ \tilde{v} \circ \Phi \qquad (1)$$

where $g : \mathbb{R} \to \mathbb{R}$ is a (non-decreasing) contrast change, $\delta(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^2$ is the disparity induced from one image to the other by the urban surface (*i.e.* proportional to the elevation map $h(\mathbf{x})$ as seen before).

This model is more and more accurate as $B/H \to 0$. Even if the images are not taken from the zenith, the small B/H ensures that no significant change of the occlusion occurs between both shots, so that they are linked by a simple "deformation" model like equation (1).

In addition, the fact that both images from the pair are taken simultaneously implies that no illumination changes occurred, so that the contrast change g can be well approximated by a linear one, simplifying equation (1):

$$v(\mathbf{x}) = (\tilde{v} \circ \Phi)(\mathbf{x}) \qquad (2)$$

This is the ideal case where B/H is extremely small. In practice we shall be dealing with larger B/H ratios where occlusion artifacts are more important. We explain how to deal with such cases in Section 3.

Sampling Now, in reality, an ideal continuous image like v or $\tilde{v}$ can only exist as a model. An actual acquisition device will blur, sample and add noise to both shots

$$u = S_1(p * v) + n \qquad \tilde{u} = S_1(p * \tilde{v}) + n \qquad (3)$$

where the noise n is usually modelled as a zero-mean Gaussian and $S_s : L^2(\mathbb{R}^2) \to \ell^2(s\mathbb{Z}^2)$ is the sampling operator on a discrete regular grid of step s. The convolution kernel p is dictated by the physical characteristics of the instrument, but we shall assume here that it is a positive kernel with small support which closely approximates the sinc kernel ($\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$) at the right scale

$$p(\mathbf{x}) \approx \mathrm{sinc}_s(\mathbf{x}) = \mathrm{sinc}(\mathbf{x}/s) \quad p \geq 0$$

typically a prolate or cardinal spline kernel. This assumption ensures (thanks to Shannon's sampling theorem) that the continuous image $p * v$ can be accurately restored from its samples u, avoiding the introduction in u of ringing or aliasing artifacts that were not present in v.

Such an assumption does not hold in general for real systems (except maybe for highly tuned very high quality systems), but a calibration and restoration procedure [2, 1] can be used to obtain such a canonical representation for

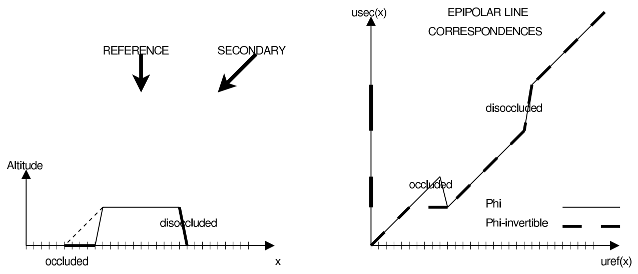

Figure 1. Problems that appear when inverting a disparity function $\delta(x)$ (left image), the pseudo-inverse of the function $\Phi(x) = x + \delta(x)$ that represents correctly the occlusions is shown in the image of the right.

any image obtained by a known device. This representation will be consistent across the different viewpoints and scales that will be simulated for an input image or image pair.

Sampling a deformed image From the equations above it turns out that in order to simulate a secondary view $\tilde{u}$ from a reference view u and the knowledge of the disparity map δ (or its associated deformation Φ) we should first deconvolve and denoise u, to obtain as close a representation as possible to the original v, and then use equation (3) to simulate $\tilde{u}$

$$\begin{aligned}\tilde{u} &\approx S_1\left[p * ((p^{-1} * u) \circ \Phi^{-1})\right] + n \\ &\approx S_1\left[p * ((\mathrm{sinc} * u) \circ \Phi^{-1})\right] + n.\end{aligned}$$

The deconvolution process is however very ill posed. In practice, a discrete Wiener deconvolution filter [2] for p will (in our canonical setting) be very close to the sinc kernel, and this is what we use in practice. Section 4 gives the numerical details on how this is implemented with a reasonable compromise between lack of artifacts, accuracy, and computational performance. A second inverse problem concerns the inversion of the deformation Φ^{-1}, which shall be dealt with in the next Section 3.

3 Simulating geometric deformations

Equation (1) can be rewritten in terms of the mapping function Φ:

$$\Phi(\mathbf{x}) = \mathbf{x} + \delta(\mathbf{x}) \qquad (4)$$

We can analyze the behaviour of this function in presence of occlusions and disocclusions (See Figure 1):

occlusion If the function $\Phi(\mathbf{x})$ is decreasing then some parts of the reference image u will not be visible in the secondary image $\tilde{u}$. When simulating $\tilde{u}$ from u we need to simulate these occlusions. Recall that function Φ is decreasing in $[a, b]$ if $\Phi(\mathbf{x})' < 0$ for all $\mathbf{x} \in (a, b)$, which leads to $\delta'(\mathbf{x}) \leq -1$.

disocclusion If $|\Phi'(\mathbf{x})| > 1$ (i.e. if $\delta'(\mathbf{x}) \geq 0$) then the sampling rate of the secondary image $\tilde{u}$ will be greater than in the reference image, which means that more detail can be represented in $\tilde{u}$ than in u. When simulating $\tilde{u}$ from u the problem here is how to "disocclude" this information missing in u.

In urban terrains (due to the ubiquity of vertical walls) non-decreasing functions are very unlikely, then the function $\Phi(\mathbf{x})$ will not be invertible. On the other hand if we want to simulate the secondary image $\tilde{u}$ from its relation to the known data u and Φ (see equations (1) and (3)), then we may need to compute its values over a regular grid

$$\tilde{u}(\mathbf{x}) \approx u(\Phi^{-1}(\mathbf{x})) \quad \text{with } \mathbf{x} \in \mathbb{Z}^2 \tag{5}$$

which means interpolating u on the irregular (perturbed) grid $\Phi^{-1}(\mathbb{Z}^2)$.

These observations lead to the need of inverting the (not always invertible) function $\Phi(x)$, and in order to solve the problem we propose to compute a *pseudo-inverse* of this function such that the occluded areas are correctly represented. An occlusion occurs when the same point in the second image (the vertical axis in Figure 1) corresponds to more than one point in the reference image (horizontal axis), the computation of the pseudo-inverse must solve this ambiguity. The solution consists in keeping the highest point value in the areas where the inverse has multiple functional values, as expressed in equation (6) and shown in Figure 1. That is, from all $\mathbf{x}$ that satisfy $\Phi(\mathbf{x}) = \mathbf{y}$, we define the pseudo-inverse Φ^+ of $\mathbf{y}$ to be the maximum of those $\mathbf{x}$:

$$\Phi^+(\mathbf{y}) = \max\{\mathbf{x} : \Phi(\mathbf{x}) = \mathbf{y}\} \tag{6}$$

Another problem related to the simulation is the disocclusion. Here we opted for the simplest solution, which does not search to fill-in any unknown information, but just interpolate and filter. This results in a blurring effect since the last known value is dragged along the disoccluded area. A more visually pleasing (but not necessarily more accurate) result would be obtain by "inpainting" disoccluded areas by iterative copy-paste techniques like the one proposed by the seminal work of Efros-Leung [4] and the variations that followed.

Simulation of altitude changes The previous discussion dealt with the simplest application scenario where we want to simulate a secondary image $\tilde{u}$ from a known reference image u and a disparity map δ. We turn now to the more complex problem of *changing* the B/H ratio of a given stereo pair u, $\tilde{u}$. The first step will be to estimate the disparity δ between both images in the pair if it is not already known.

The B/H ratio can be increased by either enlarging B or reducing H. We chose the second option here. Changing the satellite altitude introduces two modifications to the scene that we must incorporate to our model in order to obtain an accurate simulation. There is a scale change on both images by a zoom in or a zoom out, depending on whether the satellite moves up or down the scene. Second, a geometric change also takes effect, that deforms the images, even occluding or desoccluding some of the original structures present on the scene.

More precisely, and recalling equation (7), to simulate a reduction by a factor of k of the satellite's altitude H while keeping the same baseline B we simply divide the altitude by the factor we want:[1]

$$\delta'[\text{pixels}] = \frac{B}{H}\frac{1}{R}kh[\text{meters}] \tag{7}$$

Using both equations we obtain a relation between the original disparity (or deformation) values and the new ones:

$$\delta_k := \delta k \qquad \Phi_k(\mathbf{x}) := (\mathbf{x} + k\delta(\mathbf{x})) \tag{8}$$

Thus, if we have a disparity map δ with an altitude of H, the last equation states that we can obtain the disparity map with an altitude H/k simply multiplying the original disparity map by k.

Here we assume that the ideal reference image v of the stereo pair is taken from the zenith in nadir orientation. This means that an altitude change does not affect the reference image but only the secondary image $\tilde{v}_k$

$$v(\mathbf{x}) = \tilde{v}(\underbrace{\Phi(\mathbf{x})}_{\mathbf{y}}) \quad \text{and} \quad v(\mathbf{x}) = \tilde{v}_k(\underbrace{\Phi_k(\mathbf{x})}_{\mathbf{z}})$$

Note that we could proceed from the second equation as in the previous Section, by computing the pseudoinverse of Φ_k and applying it to v in order to obtain $\tilde{v}_k$. But this procedure would be based only on v, ignoring the information in $\tilde{v}$ which is closer to $\tilde{v}_k$ than v. A more sensible approach would simulate $\tilde{v}_k$ directly from $\tilde{v}$. This amounts to computing (via $\mathbf{x}$) the link between $\mathbf{z}$ and $\mathbf{y}$.

$$\begin{aligned} &\mathbf{z} - \mathbf{y} = \Phi_k(\mathbf{x}) - \Phi(\mathbf{x}) = (k-1)\delta(\mathbf{x}) \\ \Rightarrow \quad &\mathbf{z} = \mathbf{y} + (k-1)\delta(\mathbf{x}) \approx \mathbf{y} + (k-1)\delta(\mathbf{y}) = \Phi_{k-1}(\mathbf{y}) \end{aligned}$$

The last approximation is based on the fact that $\mathbf{x}$ and $\mathbf{y}$ are close, and that δ is small and regular. Note that this approximation justifies the interpolation that we use in practice, namely

$$\tilde{v}_k \approx \tilde{v}_1 \circ \Phi^+_{k-1}$$

4 Applying deformations and scale-change

The equation (5) describes how to simulate a new view $\tilde{u}$ from u by applying the pseudo-inverse $\Phi^+ : \mathbb{R}^2 \to \mathbb{R}^2$ to the original sampling of u, but this perturbed sampling is generally not equispaced. Therefore simulating $\tilde{u}$ entails an interpolation of u.

As explained in Section 2, u can be reconstructed from its samples with the sinc interpolation according to

[1] We must be careful not to use a very large reduction factor k because equation (7) relies on a parallel projection model that is valid only as long as $\frac{H}{k} \ll h$. For larger values of k more complex perspective deformations should be taken into account.

the Shannon's sampling theorem. It is customary for the interpolated function u and the kernel k, to be considered as periodic functions (of period N in each direction and sampled with step 1). So that they are determined by a finite number of Fourier coefficients (denoted $\hat{u}(\omega)$, $\mathbf{x} \in \mathbb{Z}^2 \cap [0, N)^2$ and efficiently computed with the FFT: *Fast Fourier Transform*) or by the same number of samples that are also denoted $u(\mathbf{x})$, $\mathbf{x} \in \mathbb{Z}^2 \cap [0, N)^2$. For this setup the interpolation of u reduces to the evaluation of the trigonometric polynomial:

$$u(\mathbf{x}) = \sum_{\omega \in \{0,\cdots,N-1\}^2} \hat{u}(\omega) e^{2\pi i \mathbf{x}\omega/N} \quad \mathbf{x} \in \mathbb{R}^2 \tag{9}$$

When $\mathbf{x} \in \mathbb{Z}^2$, the sampling (9) is efficiently computed by the inverse FFT using a divide and conquer approach. The FFT's approach reduces the number of floating point operations from $\mathcal{O}(N^4)$ for a naïve evaluation, to $\mathcal{O}(N^2 \log N^2)$. However, in the case of nonequispaced sampling it is no longer possible to apply the FFT's srtategy.

The NFFT (*Nonuniform Fast Fourier Transform*) [8] is a C subroutine library for computing the nonequispaced discrete Fourier transform (NDFT), that allows to approximate (9) in $\mathcal{O}(N^2 \log N^2)$. In a nutshell, the principle behind the NFFT's efficiency is its memory/computation tradeoff. NFFT pre-computes and stores a zoomed version of the signal (let's say that the zoom factor is C, then the cost is $\mathcal{O}(C^2 N^2 \log N^2)$). Then the value of each sample is approximated over the zoomed signal by a high order B-spline interpolation (with a cost comparable to the one of the FFT zoom).

When simulating a new view, the re-sampling of u leads to the apparition of alias in the simulated image $\tilde{u}$. Alias may appear when the deformation Φ^+ induces a subsampling of u, and under this circumstance the result will be an aliased texture.

A realistic simulation of $\tilde{u}$ should not present alias.

Indeed the generation of a new view involves not only the re-sampling of u but also the simulation of the entire acquisition system

$$\tilde{u}(\mathbf{x}) \approx (p * ((\text{sinc} * u) \circ \Phi^+))(\mathbf{x}) \qquad \mathbf{x} \in \mathbb{Z}^2 \tag{10}$$

where Φ^+ stands for a continuous distortion function to be applied to u.

Recall that u can be interpolated from its samples, but the samples of $\tilde{u}$ may not be sufficient to recover it because some areas of the image may not respect the Nyquist band-limit/sampling constraint any more. To adequately simulate $\tilde{u}$ we will proceed as follows: First compute a zoomed version of $u \circ \Phi^+$ so that no alias will be present in it, then apply the filter p to remove the high frequencies appeared due to the deformation and lastly subsample.

For the first step we zoom by zero-padding $u \circ \Phi^+$ using a zoom factor depending on the maximum slope of Φ^+. For simplicity suppose that this slope is 2, meaning that the samples are moving apart (like in a subsampling), then the zoom factor should be 2. Notice that the distortion map $\Phi^+(\mathbf{x})$ should be interpolated as well, this is done by a linear interpolation of the original map (6). The oversampling will provide the necessary spectral room to represent the new high frequencies that may appear due to the deformation.

The last step of the simulation involves the simulation of the acquisition system by convolving with the filter p (performed in frequency domain) –this will smooth out the frequencies responsible for the alias– followed by a sampling of a factor 2.

5 Performance-accuracy trade-offs

A real-time simulation of the stereo pair requires a fast navigation between the different altitudes, which entails a zoom for the reference image and the simulation of a new secondary image. It is clear from the previous Section that, it is not possible to accurately simulate the secondary image in real-time scenarios.

The two main bottlenecks of the schema outlined in Section 4 are the computations of FFT's, NFFT's, and pseudo-inverses. To avoid most of the FFT's cost we removed the zero-padding zooms, trading it for some minor aliasing artifacts. And to avoid the re-estimation of the pseudo-inverse we will pre-compute it at some coarse scales, and interpolate them on-line to obtain the spatial coordinates of each pixel at the target altitude.

Thus, we need not only the pre-computed disparity maps but also maps of deformed image coordinates at varying B/H ratios. These are summarized in the following preprocessing (offline) steps.

1. Given the original disparity map δ, the primary and secondary images u, $\tilde{u}$ and the original height H, divide the altitude interval to be simulated in L steps $H = H_1 > \cdots > H_L \gg 0$.
2. For each altitude H_l compute the ratio $k = \frac{H}{H_l} \geq 1$ and the corresponding rescaled disparity map $\delta_{k-1}(x) = (k-1)\delta(x)$, and the pseudo-inverse $\Phi^+_{k-1}(x)$, as in equations (6) and (8). Store its integer values in $T_l : i \mapsto \Phi^+_{k-1}(i)$, for $i \in \mathbb{Z}^2 \cap [0, N)^2$

The secondary image $\tilde{u}_{sim}$ is then simulated (on the fly) at the desired altitude H_{sim}

1. (Init) Let (H_l and H_{l+1}) be the two pre-computed maps so that: $H_{sim} = aH_l + (1-a)H_{l+1}$ with $(0 \leq a \leq 1)$
2. (Blending) The value of T_{sim} is then computed as

$$T_{sim}(i) = aT_l(i) + (1-a)T_{l+1}(i).$$

3. (Resampling) Use T_{sim} to sample the values of $\tilde{u}_{sim}$ using the equivalent of equation (10) for variable B/H:

$$\tilde{u}_{sim} = S_1[p * ((\tilde{u} * \text{sinc}) \circ T_{sim})]. \tag{11}$$

4. (Zoom) Apply to u and $\tilde{u}_{sim}$ a zoom factor of H/H_{sim} to simulate the change of altitude.

the 1st, 2nd and 4th steps have all linear complexity (linear with the number of pixels), whereas the 4th is a fast zoom that can be done by the graphic card. Step 3 is more expensive, and can be performed in the following ways (which are variants of the algorithm)

1. The highest accuracy (v1.3) for step 3 implies four steps: *(i)* a *zoom* $\times s$ of $\tilde{u}$ by zero-padding (FFT); *(ii)* zoomed *reinterpolation* to obtain $\tilde{v}_{sim}(i) := ((\tilde{u} * \text{sinc}) \circ T_{sim})(i)$ for $i \in (\mathbb{Z}/s)^2$ (NFFT); *(iii) low-pass filtering* $p * \tilde{v}_{sim}$ to avoid aliasing; *(iv) subsampling* $\tilde{u}_{sim}(i) := (p * \tilde{v}_{sim})(i)$ for $i \in \mathbb{Z}^2$. In addition we should compute $T_{sim} = \Phi^{+}_{k-1}(x)$ exactly for $k = \frac{H}{H_{sim}}$ instead of using the blending in step 2. The main performance penalty comes from the NFFT interpolation $\mathcal{O}(N^2 log N^2)$.

2. A faster, but still pretty accurate way (v1.2), will compute T_{sim} by blending, limit the zoom in step 3.i to $s = 2$, use 2nd order splines instead of the slower FFT for the zoom, low-pass-filtering and subsampling steps, and approximate the zoomed NFFT in step 3.ii by bilinear interpolation ($\tilde{v}_{sim}(i) := \left((\tilde{u} * \beta^1) \circ T_{sim}\right)(i)$). The costs of all these operations are still $\mathcal{O}(N^2)$ but with a fairly large constant.

3. The fastest way (v1.1) is to apply a linear interpolation instead of the NFFT in step 3.ii and drop the zoom, low-pass-filtering and subsampling steps completely. This option keeps the cost of the algorithm in $\mathcal{O}(N^2)$ (recall that N is the number of lines of the image). But the absence of zoom will result in the emergence of some aliasing artifacts.

v1.1		v1.2	
Steps	Seconds	Steps	Seconds
Blending	< 0.01	*Blending*	0.01
Resampling...		*Resampling...*	
		Zoom 2x	0.95
Linear Interp.	0.09	Linear Interp.	0.36
		Low-pass filt.	0.57
		Sampling	0.01
Total	0.1	Total	1.99

Table 1. Execution Times.

Table 1 gives a break-down of the execution times into steps for the proposed algorithms, when applied to a 512×512 pixels image on a Pentium 4. As we can see, both methods can be considered real-time. Although v1.2 is close to the acceptable limit for an interactive application. So, our solution consists in running both algorithms as parallel threads. v1.1 finishes first and gives a first low-quality approximation to preserve interactivity. When GUI-interaction slows down for a while v1.2 has enough time to finish and render a medium quality approximation. A third thread can also be added for the final very high quality rendering of the simulation v1.3.

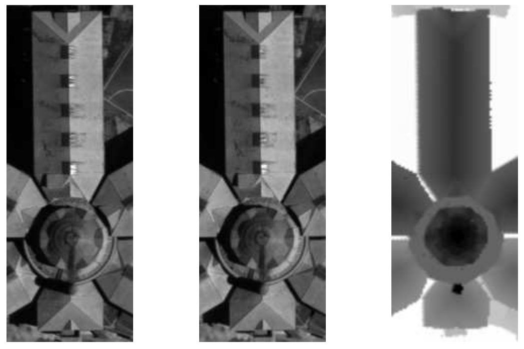
(a) Reference (b) Secondary (c) GT

Figure 2. A stereo pair, taken with a baseline/altitude ratio of $B/H = 0.045$ and the corresponding ground-truth.

I/O issues The real-time visualization tool is expected to handle large images efficiently, and the architecture should contemplate these issues. The biggest problem is that due to the size of the images, we could not load them completely into memory. Some critical I/O operations are related to: the loading of a spatial region of the image for visualization and the loading of zoomed or degraded versions of the image. These operations are fully implemented in JPEG2000 and JPIP standard [7]. Using the JPEG2000 format we have a scale- and space-indexed multiresolution representation that can be used directly to perform most of the operations described before, without the need of performing zero-padding zoom since they are efficiently precalculated and stored in JPEG2000 as wavelet coefficients.

6 Experiments

For the experiments we used a real stereo pair of aerial images taken at a B/H factor of 0.045. This 512×512 and the corresponding ground-truth was kindly provided by CNES. For display purposes in Figure 2 is shown a 125×335 crop of this data set.

To validate the accuracy of the simulated images we start by computing precalculated simulations $\tilde{u}^k$ (for $k = 2, 6, 10$) of altitude changes (H/k). The results are shown in Figure 3. In this experiment, the disocclusion areas were replaced by a linear interpolation between non-occluded adjacent points, which generates the artifact that we can see at the junction of the football field and the building.

Given the simulations obtained before, the next experiment is to compute the disparity map using a few algorithms for highly accurate subpixel disparity map computation [6]. Since we have the ground-truth, we can perform quantitative measurements of the results, that are listed in Table 2.

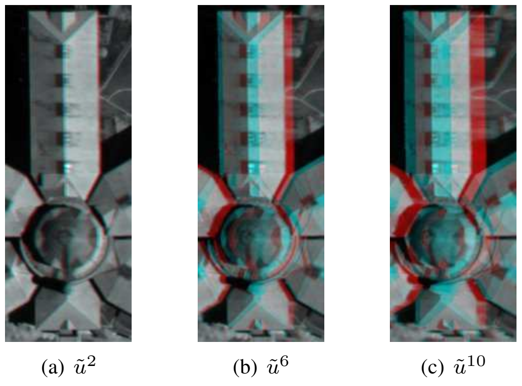

Figure 3. Precalculated reinterpolations shown as anaglyphs, using $\tilde{u}$ in the red channel and $\tilde{u}^k$ in green and blue channels.

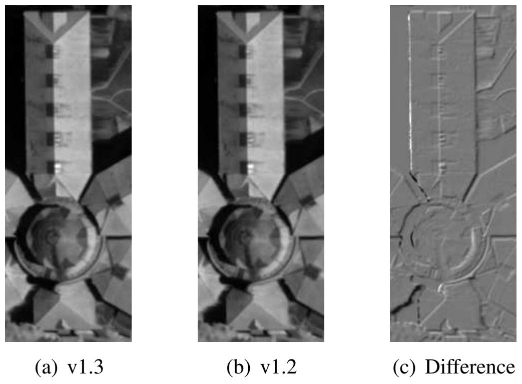

Figure 4. v1.3 vs v1.2. The images have a graylevel range $[0, 256]$ while the depicted errors have a range of $[-32, 37]$.

The most important of these results is that we can obtain better disparity maps when we use the simulated images as secondary image. This does not mean that larger baselines are better (in a real situation larger baselines imply –among other things– illumination changes that affect accuracy and that we are not simulating), it only indicates that the simulation is very accurate, since it allows to retrieve the original disparity. So our algorithm (in its most accurate version) is not only well suited for visualization, but also for building stereo-pairs and testing stereo reconstruction algorithms at several B/H settings that may not be available among the real data.

Finally, in Figures 4(a) and 4(b) we show the simulation of $\tilde{u}^{6.5}$ using v1.3 and v1.2. As we can see, both images look very similar to the naked eye. So this is an example that the real-time algorithm v1.2 is valid to simulate a navigation between the different altitudes. In order to see better the areas where the algorithm is less accurate, in Figure 4(c) we show an image that is the difference between Figures 4(a) and 4(b).

Stereo Algorithm	$\tilde{u}$	$\tilde{u}_{sim}$	$\tilde{u}^2$	$\tilde{u}^3$
MARC[3]	0.438	0.317	0.259	0.230
MERGE-NFA[6]	0.438	0.319	0.302	0.276

Table 2. Disparity Maps Errors using L_1.

7 Conclusions and Future Work

In this work we have covered all the stages needed for simulating stereographic image pairs from a given low-baseline one. We considered two different scenarios: simulating high baseline from a low-baseline pair and simulating new low-baseline pairs from an original low-baseline one. We have explained the mathematical framework needed to derive the simulation algorithms, and given the details of the overall architecture for a real-time visualization application. In both cases we have developed the algorithms and the software architecture to obtain both real-time simulations or highly accurate simulations.

Many improvements are still possible. In terms of quality, most simulation errors occur near discontinuities and disocclusions, where the deformation model is less accurate. Filling in those regions with a synthesized texture will greatly improve the visual quality of the results. Computational performance can also be greatly improved by implementing most of the time consuming operations directly on the GPU. Further improvements are also possible by optimizing the data transfers between hard-disk, RAM and GPU-RAM.

References

[1] A. Almansa, V. Caselles, G. Haro, and B. Rougé. Restoration and zoom of irregularly sampled, blurred, and noisy images. *SIAM-MMS*, 5(1):235–272, 2006.

[2] A. Almansa, S. Durand, and B. Rougé. Measuring and improving image resolution by adaptation of the reciprocal cell. *J. Math. Imaging Vision*, 21(3):235+, 2004.

[3] J. Delon and B. Rougé. Small baseline stereovision. *J. Math. Imaging and Vision*, 28(3):209–223, 2007.

[4] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV'99*, vol. 2, 1999.

[5] O. Faugeras, Q.-T. Luong, and T. Papadopoulo. *The Geometry of Multiple Images*. MIT Press, 2001.

[6] L. Igual, J. Preciozzi, L. Garrido, A. Almansa, V. Caselles, and B. Rougé. Automatic low baseline stereo in urban areas. *AIMS - IPI*, 1(2), 2007.

[7] JPEG. Jpeg 2000 part i final committee draft version 1.0. Technical report, Joint Photographic Experts Group, 2000.

[8] D. Potts, G. Steidl, and M. Tasche. FFT for nonequispaced data. In J. Benedetto and P. Ferreira (Eds.), *Modern Sampling Theory: Mathematics and Applications*, ch. 12. Birkhäuser, 2001.

[9] B. Rougé. Théorie de la chaîne image optique et restauration à bruit final fixé. HDR, 1997.