EG UK Computer Graphics & Visual Computing (2014)
Rita Borgo, Wen Tang (Editors)

Modelling of Clouds from a Hemispherical Image

Thiemo Alldieck[1], Dennis H. Lundtoft[1], Niels Montanari[2], Ivan Nikolov[1], Iskren G. Vlaykov[1], Claus B. Madsen[1]

[1]Aalborg University
[2]Bordeaux Institute of Technology

Abstract

This paper presents an image-based method for modelling clouds. Unlike previous image-based approaches, a hemispherical photograph is used as input, enabling to consider an entire sky instead of merely a portion. Our method computes the intensity and opacity of the clouds from the photograph. For this purpose, beforehand, the sun illumination is filtered, the pixels are classified between cloud and sky pixels, and the sky behind the clouds is reconstructed. After having been smoothed, the intensity of the clouds is used to create vertices on a hemisphere, and their radius coordinate is modulated by the intensity value of the corresponding pixel. Finally, the mesh is generated by triangulation of the vertices. Additionally, the use of the opacity of the clouds to simulate their transparency and render them is proposed. The results show that our method can be used to produce a realistic full sky populated with clouds in a very straightforward way for the user.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction and Previous Work

Clouds are typical elements in image synthesis of outdoor scenes. They are displayed in a variety of graphical applications such as flight simulators, meteorological visualisation software, films and video games. In this paper, we focus on modelling of clouds. A simple method to simulate the transparency of the generated clouds is proposed in addition. Approaches to modelling clouds may be classified into three categories: procedural, physics-based and image-based.

The procedural methods can synthesise clouds with a low computational cost. However, adjusting the parameters by trial and error process is often necessary to obtain clouds with an acceptable level of realism. The combination of pseudo-randomness such as fractal noise [Per85] and implicit surfaces such as metaballs is one classical approach to procedurally modelling clouds [Elb97, BN04]. A sketch-based interface aiming at avoiding a long trial and error process and quickly modelling cumulous clouds has been proposed [WBC08].

The physics-based methods rely on atmospheric fluid dynamics to simulate the physical process of cloud formation [MYND01, HBSL03]. While the clouds are synthesised with a high level of realism, the computational cost is high too. Furthermore, adjusting the physical parameters in order to get the desired shape may be hard for artists. To address this issue, a method for controlling the simulation to generate desired shapes of cumulus clouds has been proposed [DKNY08].

Lastly, the image-based methods extract information from photographs to create clouds visually similar to those of the photographs. This approach to model clouds has been less investigated than the procedural and physics-based approaches. Satellite images have been used for modelling earth-scale clouds viewed from space [DNYO99, YLHY13]. Dobashi et al. proposed a method for modelling cirrus, altocumulus and cumulus clouds from a photograph [DYY10]. Based on this work, an alternative method targeting cumulus clouds has been presented recently [YLH*14].

Our method is an image-based method for modelling clouds. Instead of a usual photograph, a hemispherical photograph is used as input, enabling to consider an entire sky instead of merely a portion. Our method presents some similarities with Dobashi et al.'s method [DYY10]: both aim at computing the intensity and opacity of clouds from the photograph, so as to use it for generating the cloud model; a classification between cloud and sky pixels as well as a sky reconstruction are common steps of this computation. However, taking into account the presence of the sun in an entire sky image, a preliminary step is necessary to filter the sun il-

DOI: 10.2312/cgvc.20141202

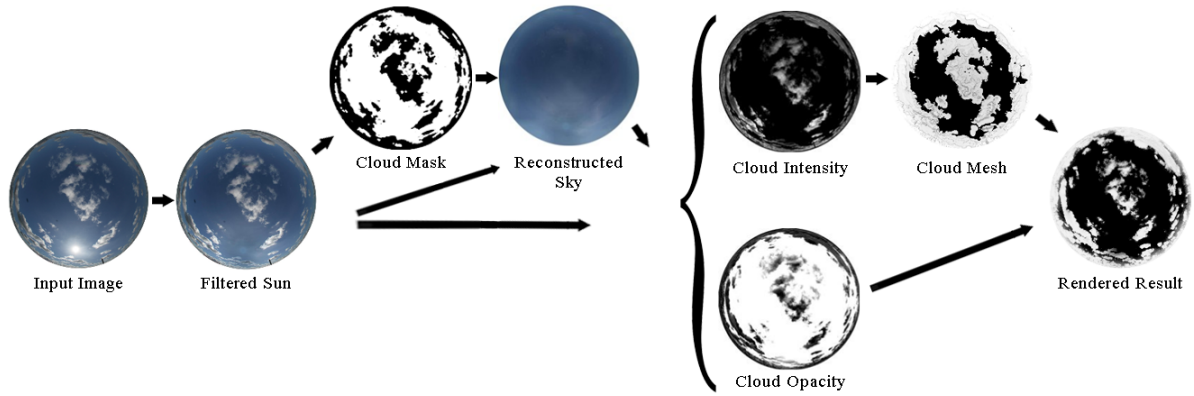


Figure 1: *Overview of the method, displaying the result images of the different components and their dependencies*

lumination. Furthermore, several improvements and adaptations are realised: the calculation of the intensity and opacity of the clouds has been made simpler; the classification step has been enhanced; and an alternative sky reconstruction technique, less sensible to intrusive objects, such as buildings at the horizon, and which deals better with the images containing a large amount of cloud, is employed. Since it borrows from [DYY10] the techniques to classify the pixels into either a cloud or a sky pixel as well as to reconstruct the sky, [YLH*14] could also benefit from our improvements.

Our method generates a realistic entire sky populated with clouds that are visually similar to those from the hemispherical photograph. Since the clouds are based on real ones of a same sky, they are ensured to be consistent with each other, and so the realism of the produced sky is improved. Using our method, a full sky populated with clouds is produced in a very straightforward way for the user. Furthermore, contrary to directly using the photograph, using synthesised clouds enables them to be rendered under different lighting conditions or to be used in combination with a non-photorealistic rendering method.

2. Method

2.1. Overview of the method

Our method uses a hemispherical photograph of an entire sky as input. The first steps are digital image processing computations aiming at extracting relevant items of information from the photograph. Afterwards, in the further steps, this information is used for creating the surface shape of the clouds. The pipeline of the overall algorithm is illustrated in figure 1.

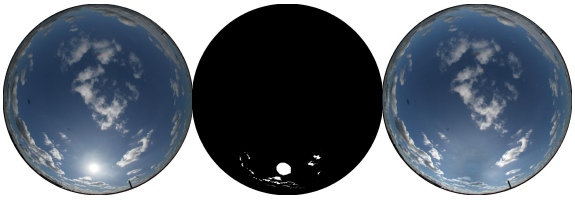
The items the first steps are aiming at are the intensity and opacity of the clouds, calculated at each pixel. This calculation requires the background image of the sky in addition to the input photograph. It is thus necessary to estimate the sky colours behind the clouds, and so beforehand to classify the pixels into either a cloud or a sky pixel. Finally, because the sun makes its neighbourhood brighter on the photograph, an initial step must be applied to filter the sun illumination, so as to recover correct colours at the surrounding pixels. Our method performs these steps in reverse order to finally lead to the intensity and opacity of the clouds.

Then, the intensity of the clouds is used for the modelling. In order to reduce the sharpness of the cloud shape, the intensity is smoothed beforehand. Vertices are created on a hemisphere, and their radius coordinate is modulated by the value of the corresponding pixels of the intensity map. Finally, the mesh is generated by triangulation of the vertices. In addition, a simple method based on the opacity of the clouds is used for the rendering.

2.2. Filtering of the sun illumination

The initial step of the method is the filtering of the sun illumination. First, the position of the sun and the radius of its saturated halo are computed. Using these items of information, a function representing the sun illumination is applied to each pixel. The resulting image is then subtracted from the input image, leading to an image with the sun illumination filtered.

In order to compute the sun position and the radius of its saturated halo, in number of pixels, our method starts by creating a greyscale image from the input image. A binarisation of the greyscale image is performed. The user must adjust the threshold of the binarisation ε_s so that the contour of the saturated halo of the sun is displayed well-defined in the binarised image. The saturated halo of the sun is assumed to be the blob with the biggest size. Then, successive morphological erosions with a same structuring element of small size are applied to the binarised image. At each iteration of the erosion process, the previous image is stored in an allo-



(a) (b) (c)

Figure 2: ***Filtering of the sun illumination.*** *(a) Input hemispherical image; (b) Binarised image; (c) Image with the sun illumination filtered.*

cated buffer. The process ends when the image is completely eroded. Afterwards, the saturated halo radius is calculated from the size of the structuring element and the number of iterations. The sun position coordinates are calculated from the previous image by taking the mean value of each coordinate of the non-eroded pixels.

To represent the sun illumination at its neighbourhood, two-dimensional circular Gaussian functions centred on the sun position are used. With the Gaussian parameters A and σ and in a polar coordinate system centred on the sun centre, their general form is:

$$f(r) = Ae^{-\frac{r^2}{2\sigma^2}} \tag{1}$$

We make A varying with the colour component λ, equals to either red R, green G or blue B. In addition, we link $A(\lambda)$ and σ to the sun intensity $I_{sun}(\lambda)$ and the saturated halo radius r_{sun}, respectively, and introduce new parameters $\alpha(\lambda)$ and β such that $A(\lambda) = \alpha(\lambda)I_{sun}(\lambda)$ and $\sigma = \beta r_{sun}$. It enables to avoid the need to readjust the parameters if the intensity scale or the image resolution varies. For each pixel p, we also multiply the Gaussian function by the value in the input image $I(p)$. Finally, the generalised form of the function representing the sun illumination is:

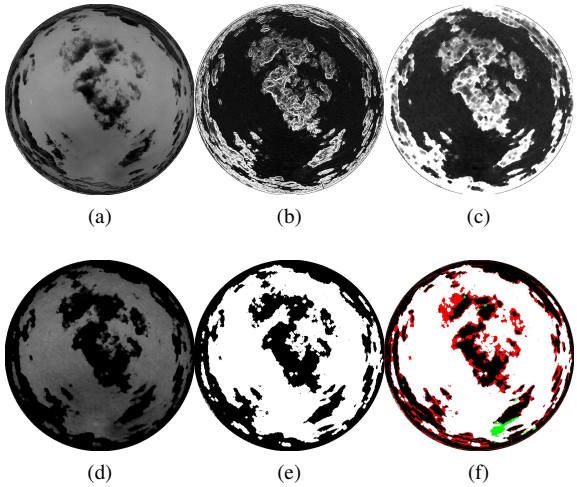
$$f(r,\lambda) = \alpha(\lambda)I_{sun}(\lambda)e^{-\frac{r^2}{2(\beta r_{sun})^2}}I(p,\lambda) \tag{2}$$

The resulting image is subtracted from the input image. Figure 2 illustrates the process of the filtering of the sun illumination.

2.3. Classification between cloud and sky

The pixels of the image obtained after the filtering of the sun illumination step are classified into either a cloud or a sky pixel. For each pixel p, for which the intensity is $I(p,\lambda)$, the saturation $S(p)$ is calculated using the formula below, as defined in [DYY10]:

$$S(p) = \frac{I_{max}(p) - I_{min}(p)}{I_{max}(p)} \tag{3}$$



(a) (b) (c)

(d) (e) (f)

Figure 3: ***Classification between cloud and sky.*** *(a) Saturation image; (b) Gradient information image with multiplier adjustment; (c) Gradient information image with multiplier adjustment and morphological closing; (d) Image resulting from the combination of the saturation and the gradient information; (e) Binarised image; (f) Comparison image between Dobashi et al.'s method and ours: the pixels classified as cloud only by Dobashi et al.'s method or only by ours are displayed as green and red, respectively.*

where

$$\begin{cases} I_{max}(p) = \max\limits_{\lambda\in\{R,G,B\}}(I(p,\lambda)) \\ I_{min}(p) = \min\limits_{\lambda\in\{R,G,B\}}(I(p,\lambda)) \end{cases}$$

The clouds are assumed to be white or grey, with little difference between the values of the different colour components; thus, the saturation of a cloud pixel is expected to be small. On the other hand, the sky is assumed to be blue, with a significant difference between the values of the red and green colour components and the value of the blue component; thus, the saturation of a sky pixel is expected to be high. In Dobashi et al.'s method [DYY10], the classification of the pixels is directly done by binarisation of the saturation image. However, the areas containing thin clouds, notably the border of the clouds, have a not so high saturation level, and the classification of these regions tends to fail. To address this issue, in our method, a gradient information is used in combination with the saturation. The gradient being especially high at the cloud borders, it is used to compensate the relatively low saturation of these regions. Given a pixel p, for which the intensity is $I(p)$, the gradient information value $G(p)$ is calculated with the following formula:

$$G(p) = \frac{||\overrightarrow{grad}(I(p))||_2 - grad_{min}}{grad_{max}} \tag{4}$$

where

$$\begin{cases} grad_{max} = \max\limits_{k\in\{pixels\}} (||\overrightarrow{grad}(I(k))||_2) \\ grad_{min} = \min\limits_{k\in\{pixels\}} (||\overrightarrow{grad}(I(k))||_2) \end{cases}$$

Next, a morphological closing with a structuring element of small size s_c is applied to the gradient information image, so as to slightly extend the high gradient information value from the cloud borders to interior cloud parts.

Lastly, the saturation and the gradient information are combined. The weight of the gradient information is controlled by a multiplier parameter m_g, adjusted by the user. For a given pixel p, the value $V(p)$ resulting from this combination is calculated as below:

$$V(p) = S(p) - m_g G(p) \tag{5}$$

The classification is finalised by binarisation of the image resulting from the combination. The binarisation is done by setting the pixels as white if they are not completely black. The overall classification process is illustrated in figure 3.

2.4. Reconstruction of the sky

The background image of the sky is created by reconstructing the sky behind the clouds of the image for which the sun illumination has been filtered. While in the Dobashi et al.'s method [DYY10] the sky is reconstructed by solving a Poisson equation for each cloud pixel [PGB03], this technique is very sensible to intrusive objects, such as buildings at the horizon, and presents poorer results when the input photograph is populated with clouds to a large extent. Our method uses a different technique to palliate this issue. It consists of two steps. First, a sky colour for the cloud pixels of a grid is computed by interpolating the colour of a given number of nearest sky pixels. Next, a sky colour for the other cloud pixels is computed by bilinear interpolation of the colour of reference pixels, being pixels at the points of the grid.

The sky reconstruction algorithm starts by computing and storing the number of sky pixels and their coordinates. Then, a grid of the image size is created, its tiles being of a same size equals to a specified number of pixels N_t. For each pixel of the binarised image corresponding to a point of the grid, the label of the pixel is checked. If the pixel is classified as a cloud pixel, a sky colour is computed. For this purpose, the distances between the pixel and all the sky pixels are calculated, and the sky pixels are sorted according to their distance to the pixel. The contribution of the different sky pixels to the value of the sky colour of a given pixel is represented by a weighting using the inverse of the distances to the pixel. A number of nearest sky pixels N_n to be used to calculate the sky colour is specified. For a given pixel p, for which the intensity is $I(p,\lambda)$ and the distance to a sky pixel k is $d(p,k)$, the sky colour is calculated with the formula

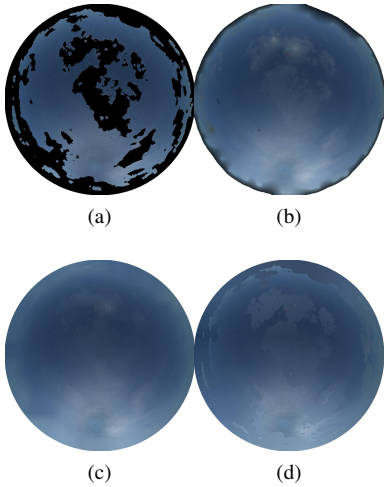


Figure 4: ***Sky reconstruction.*** *(a) Input image with the sun filtered and the clouds masked; (b) Sky image with the reconstruction processed by Dobashi et al.'s method; (c) Sky image with our reconstruction processed using $N_t = 50$; (d) Sky image with our reconstruction processed using $N_t = 200$.*

below:

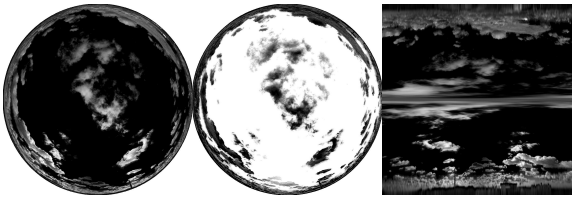
$$I(p,\lambda) = \sum_{k=1}^{N_n} \frac{w(p,k)I(k,\lambda)}{\sum_{k=1}^{N_n} w(p,k)} \tag{6}$$

where

$$w(p,k) = \frac{1}{d(p,k)}$$

The last step of the sky reconstruction algorithm aims at computing a sky colour for all the other cloud pixels. A bilinear interpolation is performed using the colour of the surrounding reference pixels, being pixels at points of the grid and the ones which delimit the tile where the considered pixel is located. The reference pixels are weighted according to their distance relatively to the pixel.

Considering a square image containing N pixels, the previously defined tile size N_t and number of nearest sky pixels N_n, the worst-case time complexity of the sky reconstruction algorithm is $O(\frac{N}{N_t^2}(N\log(N) + N_n) = O(\frac{N^2}{N_t^2}\log(N))$ if we assume the sorting algorithm to be linearithmic. The tile size ranging from 1 to $N^{\frac{1}{2}}$, the complexity varies from $O(N^2\log(N))$ to $O(N\log(N))$. Thus, N_t is a key parameter of the sky reconstruction algorithm to adjust the compromise between time-consumption and accuracy. Figure 4 presents results of sky image using different values for the tile size.



(a) (b) (c)

Figure 5: ***Intensity and opacity of the clouds.*** *(a) Intensity image; (b) Opacity image; (c) Intensity image with smoothing. (b) Intensity image with constant longitude and colatitude remapping.*

2.5. Calculation of the intensity and opacity

The effects of the sky intensity in the background have to be eliminated in order to get the intensity and opacity of the clouds. For this reason, the background image of the sky, in addition to the image filtered of the sun illumination, is needed. For each pixel p, its intensity $\gamma(p)$ and opacity $\delta(p)$ are calculated by solving the following system of equations:

$$I(p,\lambda) = \gamma(p) I_{sun}(\lambda) + \delta(p) I_{sky}(p,\lambda) \tag{7}$$

Explanations about these equations are available in [DYY10]. In Dobashi et al.'s method [DYY10], the sun is assumed to not be present in the photograph, and thus three additional unknowns, $I_{sun}(R)$, $I_{sun}(G)$ and $I_{sun}(B)$, make the system of equations unsolvable without using tricks and approximations. In the case of a hemispherical photograph displaying an entire sky, the system of equations is easily solvable if the sun is assumed not being hidden by clouds. The sun intensity is obtained from the input photograph by taking the value at the sun centre coordinates; alternatively, the sun intensity may simply be assumed to be pure white. The sky intensity is obtained from the background image of the sky. Figures 5a and 5b displays intensity and opacity image results.

2.6. Smoothing of the intensity

The intensity of the clouds is used to simulate the thickness of the clouds: the thick cloud parts are represented by a high intensity, and the thin cloud parts by a low intensity. This is used to create the mesh representing the cloud shape. However, local significant differences between cloud pixel values make the cloud shape to be visually quite unrealistic, with sharp peak and hollow patterns. In addition, because the intensity of the sky pixels is zero, the edges of the clouds tend to be abrupt. To address this issue, the values of the intensity image are smoothed using two different techniques. Depending whether the pixel is a cloud or a sky pixel, one is chosen and performed.

For each pixel of the intensity image, the value is checked. If the pixel value is zero, corresponding to a sky pixel, the new value is calculated by a weighted interpolation of the side and corner pixels, the weight of the corner pixels being half of that of the side pixels. If the pixel value is not zero, corresponding to a cloud pixel, the new value is calculated by taking the mean value between the current value of the pixel and the value obtained from the bilinear interpolation of the four side pixels.

2.7. Creation of the mesh

After having been smoothed, another transformation must be performed to the intensity image. The goal being to map the intensity image to a hemisphere, considering a spherical coordinate system, it is beneficial to create a transformed intensity image that presents constant differences of longitude and colatitude between two neighbour pixels. As a result, while the middle part have a better resolution than the border part in the original intensity image, the resolution is uniform in the transformed intensity image. The process of the transformation starts by calculating a corresponding longitude and colatitude for each pixel, so that all the latitude and longitude values differ each other by a constant step and represent a full hemisphere. For a given pixel of coordinates $(x,y) \in [0,L]^2$ in the transformed intensity image, the longitude $\theta \in [0,\pi]$ and colatitude $\varphi \in [-\frac{\pi}{2},\frac{\pi}{2}]$ are calculated by the following formulas:

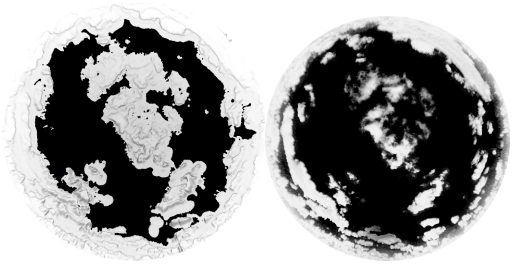
$$\theta = \frac{\pi}{L}x \text{ and } \varphi = \frac{\pi}{L}y - \frac{\pi}{2} \tag{8}$$

Then, for the point of a hemisphere corresponding to these longitude and colatitude, the Cartesian coordinates (x',y') of its projection on the basis of the hemisphere are calculated by the formulas below:

$$x' = \frac{L}{2}(\sin(\theta)\sin(\varphi)+1) \text{ and } y' = \frac{L}{2}(\cos(\theta)\sin(\varphi)+1) \tag{9}$$

Finally, the four pixels of the original intensity image for which the coordinates correspond to the floor and ceil integer values of the previously calculated coordinates are used. The value of the pixel in the transformed intensity image is calculated by a bilinear interpolation of these four pixel values, weighted accordingly to their distance relatively to the pixel. Figure 5c displays a result of the transformation of the intensity. Interpolation artefacts can be seen at the borders of the image.

The transformed intensity image is then mapped to a hemisphere of radius r. For each pixel, the corresponding colatitude is checked. If the value is not included inside a specified range $[-\varphi_l,\varphi_l]$, the pixel is discarded, in order to eliminate the most visible interpolation artefacts. Otherwise, the intensity value is checked. If it is zero, corresponding to a sky pixel, the pixel is discarded. In the other case, a vertex is created on the hemisphere, and the intensity value is used



(a) (b)

Figure 6: ***Generation and shading of the cloud model.*** *(a) Generated mesh image; (b) Generated mesh image with the transparency simulated by the opacity.*

Table 1: *Parameters*

Name	Description	Value
α	Gaussian alpha parameter	$(0.6, 0.5, 0.4)$
β	Gaussian beta parameter	2.0
ε_s	Sun detection threshold	0.98
s_c	Closing structuring element size	6
m_g	Gradient information multiplier	12
N_t	Tile size	50
N_n	Nearest pixel number	100
φ_l	Limit colatitude	12°
τ	Displacement rate	0.05

to modulate its radius coordinate accordingly to a displacement rate τ. Its Cartesian coordinates (X,Y,Z) are calculated from the intensity $I(x,y)$, (x,y) being the coordinates of the pixel, by the following formulas:

$$\begin{cases} X = (1-\tau I(x,y))r\cos(\theta)\sin(\varphi) \\ Y = (1-\tau I(x,y))r\sin(\theta)\sin(\varphi) \\ Z = (1-\tau I(x,y))r\cos(\varphi) \end{cases} \tag{10}$$

Then, for a kept pixel of the transformed intensity image, if the next pixel along one axis and the next pixel along the other one neither have been discarded, the three corresponding vertices are triangulated. The mesh is generated by repeating this process for each vertex.

Finally, the cloud shape is made two-sided by generating the mirrored mesh with respect to the hemisphere. The Cartesian coordinates of the vertices are calculated by modifying the sign $-$ by $+$ in formulas 10. A generated mesh image is shown in figure 6a.

2.8. Rendering

The rendering of the cloud mesh is based on the use of the opacity. The alpha component of the RGBA colour of the fragments to be rendered is calculated from the opacity of the corresponding pixel. For each fragment, the corresponding pixel coordinates in the opacity image are transformed using the same process that is performed for the intensity image during the creation of the mesh step, with equations 8 and 9. The value at the new coordinates in the opacity image is used as alpha component. Finally, the usual alpha blending is used to simulate the transparency of the clouds. Figure 6b displays a result image.

3. Results and Discussion

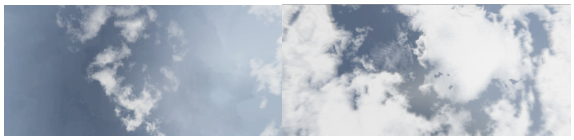
Figures 2 to 6 show the results of our method step-by-step for a same input photograph with a resolution 1000×1000 pixels. The chosen photograph presents the following features: a bright sky, a bright sun not hidden by clouds but with clouds at its surroundings, thin and thick clouds, clouds in the centre and at the borders, both the clouds and sky covering around half of the image. It has been chosen so as to correspond to an average situation. The specific parameters we used are indicated by table 1, unless otherwise specified.

Figure 3f shows the difference between Dobashi et al's classification and ours. Compared to the former, the latter is able to both recognise the thin cloud parts and avoid misclassifying the light blue sky parts. A weak point of our method is that the optimal value for the multiplier parameter m_g differs according to the distance to the centre of the image. Adjusting the parameter to fit the middle of the image leads to many sky pixels at the borders misclassified as cloud pixels. Making it slightly decreasing with the distance to the centre could solve the problem.

Figure 4b shows that the sky reconstructed by Dobashi et al's method presents black spots at the horizon, due to buildings and trees present in the photograph. Such artefacts are not noticeable in the images obtained by our mehod. Moreover, the result is clearly smoother than with Dobashi et al.'s method if the tile size N_t is small enough, such as in figure 4c. The computation durations to generate figures 4b and 4c are almost identical. An optimisation of our reconstruction would be to use an unordered partial sorting instead of a complete sorting, since it does not actually matter if the N_n nearest sky pixels are sorted or not.

We rendered the scene using the Lambertian reflection and the Gouraud shading as illumination model and interpolation technique, respectively. The reconstructed sky image is mapped to a hemisphere of higher radius than the one that shapes the cloud model. The final results, displayed in figures 7, 8 and 9, show synthesised clouds visually similar to those of the photographs.

We used an Intel i7-3630QM 2.40GHz CPU to generate the results. No multi-core nor GPU acceleration has been



(a) (b)

Figure 7: ***Result images.*** *(a) Rendered image of the generated cloud model with the reconstructed sky in the background; (b) Rendered image of another cloud model, generated from a photograph with a larger amount of cloud.*

used to speed up the computations. While the other parameters have almost no consequence on the time-consumption of our method, the tile size N_t has a significant impact. Using $N_t = 50$ for the shown images, the mesh was generated in 2min 10s. The extreme cases $N_t = 1$ and $N_t = 1000$ lead to a duration of 3h 15min and 1min 50s, respectively.

An inconvenience is that the hemispherical photographs do not have uniform level of detail, the middle being much more detailed than the borders. Therefore, the generated clouds also present a non-uniform level of detail. Notably, close to the horizon, the clouds have a very low level of detail and interpolation artefacts are clearly visible . We estimate that the amount of details might be considered as acceptable for a field of view corresponding to a colatitude φ in $[-75^\circ, 75^\circ]$.

The case of a photograph with the sun not isolated from the clouds nor hidden by clouds is not well handled by our method. In this situation, to not filter the sun illumination would involve an important part of the sky to be recognised as cloud at the classification step. However, the filtering of the sun illumination might actually add more noise into the image than it removes, making sky parts recognised as cloud and cloud parts recognised as sky.

Our method presents poorer results for photographs taken in the sunrise and sunset times. Because the sky is darker and atmospheric optical phenomena typically make the sun halo red or orange, the classification tends to fail for many pixels. However, photographs taken in the day-time, with a bright blue sky and white or grey clouds, are well handled and lead to good results.

4. Conclusion

In this paper, we have presented a novel image-based method for modelling clouds. Concretely, the contributions of our work are:

- the first method, to our knowledge, to model clouds from a hemispherical photograph, which enables to generate an entire sky populated with clouds in a very straightforward way for the user;
- a method to filter the sun local illumination, represented using a circular Gaussian function;
- an enhanced method for the classification between cloud and sky pixels, combining the saturation with a gradient information;
- an alternative algorithm for reconstructing the sky, based on a two-step interpolation for which the balance between time-consumption and accuracy is easily adjustable;
- a model for representing a sky populated with clouds, consisting of a two-sided mesh shaped around a hemisphere, for which the radius coordinate of its vertices is modulated using the intensity of the clouds;
- a simple method to simulate the transparency of the clouds generated by our method, using the opacity of the clouds.

As future work, methods to enhance the shape of the clouds at the horizon could be investigated. Adjusting the parameters α and β being not intuitive, automatizing this process would reduce the time spent in manual interaction to make the method working correctly. Also, using several hemispherical photographs, items of information relevant for animating the clouds, such as the wind strength and direction, could be extracted from the images.

References

[BN04] BOUTHORS A., NEYRET F.: Modeling clouds shape. In *Proc. of Eurographics 2004* (2004). 1

[DKNY08] DOBASHI Y., KUSUMOTO K., NISHITA T., YAMAMOTO T.: Feedback control of cumuliform cloud formation based on computational fluid dynamics. *ACM Transactions on Graphics 27*, 3 (2008), 94:1–94:8. 1

[DNYO99] DOBASHI Y., NISHITA T., YAMASHITA H., OKITA T.: Using metaballs to modeling and animate clouds from satellite images. *The Visual Computer 15*, 9 (1999), 471–482. 1

[DYY10] DOBASHI Y., YUSUKE S., YAMAMOTO T.: Modeling of clouds from a single photograph. *Computer Graphics Forum 29*, 7 (2010), 2083–2090. 1, 2, 3, 4, 5

[Elb97] ELBERT D. S.: Volumetric modeling with implicit functions: a cloud is born. In *Visual Proc. ACM SIGGRAPH 1997* (1997), p. 147. 1

[HBSL03] HARRIS M., BAXTER W. V., SCHEUEMANN T., LASTRA A.: Simulation of cloud dynamics on graphics hardware. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS Conf. on Graphics hardware* (2003), pp. 92–101. 1

[MYND01] MIYAZAKI R., YOSHIDA S., NISHITA T., DOBASHI Y.: A method for modeling clouds based on atmospheric fluid dynamics. In *Proc. of the 9th Pacific Conf. on Computer Graphics and Applications* (2001), pp. 363–372. 1

[Per85] PERLIN K.: An image synthesizer. *ACM SIGGRAPH Computer Graphics 19*, 3 (1985), 287–296. 1

[PGB03] PREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Transactions on Graphics 22*, 3 (2003), 313–318. 4

[WBC08] WITHER J., BOUTHORS A., CANI M.-P.: Rapid sketch modeling of clouds. In *Proc. of the Fifth Eurographics Conf. on Sketch-Based Interfaces and Modeling* (2008), pp. 113–118. 1

[YLH*14] YUAN C., LIANG X., HAO S., QI Y., ZHAO Q.: Modelling cumulus cloud shape from a single image. *Computer Graphics Forum* (2014), doi: 10.1111/cgf.12350. 1, 2

[YLHY13] YUAN C., LIANG X., HAO S., YANG G.: Modeling large scale clouds from satellite images. In *Proc. of the 21st Pacific Conf. on Computer Graphics and Applications* (2013), pp. 47–52. 1

(a) (b)

Figure 8: ***Comparison between:*** *(a) Clouds rendered under day-time lighting conditions; (b) Same clouds rendered under night-time lighting conditions.*

(a) (b)

Figure 9: ***Comparison between:*** *(a) Clouds from the photograph, with a resolution* 2730×2730 *pixels; (b) Same clouds generated and rendered by our method.*