

Publishers' page

Publishers' page

Publishers' page

Publishers' page

Contents

11. A rate distortion theoretic approach to remote visualization of 3D models	1
<i>Nicola Brusco, Pietro Zanuttigh, David S. Taubman and Guido M. Cortelazzo</i>	
11.1 Introduction	1
11.2 The basic RV-RD framework	5
11.3 Rendering from a single view by affine warping	8
11.4 Rendering from multiple views by DWT analysis, warping and DWT synthesis	10
11.5 The client policy	14
11.6 How to compute the distortion contribution ?	17
11.6.1 Effects of the affine warping	18
11.6.2 Effects of analysis and synthesis	19
11.6.3 Effects of extra-band energy	20
11.6.4 Effects of geometry	23
11.7 Experimental results	27
11.8 Summary	31
<i>Bibliography</i>	33

Chapter 11

A rate distortion theoretic approach to remote visualization of 3D models

Nicola Brusco, Pietro Zanuttigh, David S. Taubman and Guido M. Cortelazzo

11.1 Introduction

Remote visualization of 3D models is a new problem posing many challenging conceptual and practical questions. A basic issue addressed in some preliminary studies [Cheng and Basu (2004)] is the following: assuming one has a 3D model and may only use a fixed amount of data for it, how should he distribute the data between texture and geometry for the best rendering results? Current practical solutions for remote visualization of 3D models somehow resent the limited theoretical understanding of this issue. Indeed there exist essentially two approaches, both plagued by a considerable number of shortcomings which are worth recalling.

In the common approach to remote visualization of 3D models, which will be called for simplicity RV-CA, the server sends the 3D model geometry and its texture at the same time to the client. After transmission, the user at the client side has all the information needed in order to visualize the 3D environment and can freely navigate, even disconnecting from the server. Solution RV-CA, found in all commercial VRML browsers, suffers some notorious drawbacks. There is a latency in the visualization: since the 3D model must be completely transmitted before rendering at the client may begin, there is an unpredictable delay between the user request and the beginning of visualization. It does not take into account user interaction: all 3D data are sent, while the user might only want to navigate some parts of the 3D environment and ignore some others. The connection between

server and client is badly used: after a burst of data sent at the beginning, the connection is never used again. The client resources are not taken into account: the model could be too large to be smoothly rendered by the client.

The dual approach, which consists in not transmitting the 3D model at all, but in keeping it at server side and transmitting only the views the user wants to inspect, has also been considered in the literature as reported in Chapter 10. This approach, which will be called RV-I for simplicity, turns remote visualization of 3D models into the transmission of views upon user's demand. Since subsequent views are closely related, one natural way to improve efficiency of system RV-I is to predict each new view from the views which have already been transmitted, forming the same prediction both at server and client so that only the prediction residual needs to be transmitted. Solution RV-I is interesting as, in principle, it can be free from the drawbacks of RV-CA. Indeed, the visualization at client side can begin right away, since all it takes is the transmission of the images prompted by the user (which can happen in real time, provided adequate bandwidth is available). Data transmission is completely based upon the user requests: no useless data are sent. The connection resources between client and server are evenly used. Approach RV-I can suit the computation and memory resources of any client since it just needs to visualize a sequence of images. Furthermore, copyright can be preserved, since the user never obtains the full 3D model but only images of it. However, the intrinsic limitations of RV-I are worth pointing out. In principle the sequence of views prompted by the user could be compressed at a much lower bit rate than a standard video-sequence. In practice how to exploit the fact that the views to be transmitted all come from the same 3D model is a difficult problem, barely explored yet. In the lack of effective solutions it is safe to allocate video-rates for the implementation of RV-I and this requires considerable bandwidth resources. In any case it remains rather difficult, if not impossible, to combine information from several similar yet different views in order to synthesize a new, high quality image at a subsequent time. Such a difficulty to reuse the received information to synthesize new views at a later time corresponds to a considerable overall waste of resources. Moreover, the predictive approach can only deliver approximate representation of each view requested by the user, no matter how close those views may be to each other. Finally, system RV-I is strongly conditioned by the network: if the bandwidth decreases, the quality of the remote rendering can only decrease. If connection went down, navigating would not be possible

anymore. Indeed, the server must precisely replicate the steps performed by the client to render each new view from the previous ones.

Some hybrid solutions have been explored in order to reduce the drawbacks of RV-CA and RV-I. In [Levoy (1995)] [Koller *et al.* (2004)] the server sends a low resolution version of the 3D model to the client which the client can autonomously display locally using a graphic engine. Such a low resolution rendering is essentially meant to be used for rapid navigation. As navigation slow-downs or stops, the transmitter sends the difference between the rendering of the high resolution model available at server side and the low resolution image produced by the client. The rationale behind this hybrid approach is that the details of a 3D model are hard to perceive during fast navigation.

What we present in this chapter can be regarded as a third approach for the remote visualization of 3D models, for simplicity called RV-RD, which rests upon a new rate distortion theoretical approach. It can also be regarded as an instrument for the experimental investigation of how to subdivide geometry and texture information within a predefined data quantity in order to achieve the best rendering results, indeed, as it will be seen, this operation is deeply connected to such a question and its answer.

System RV-RD satisfies the following requirements:

- (1) Visualization at client side should start without delay, as soon as a limited amount of data is sent by the server.
- (2) The transmitted data should be based upon the user's needs: no useless data should be sent.
- (3) Data transmission should be based on the available bandwidth, in order to avoid navigation delay.
- (4) The server must have a policy to decide the kind of data most useful to send (just texture, just geometry or a suitable balance of both) and how to assign them given the available bandwidth.
- (5) The connection between server and client should be used at its full capacity.
- (6) The client must store the 3D information it receives, or part of it, according to its memory capacity.
- (7) The client must have a policy to render the 3D model, using the available data in the best way it can.
- (8) The client should be able to navigate at least some parts of the 3D environment, even if the connection with the server is down.

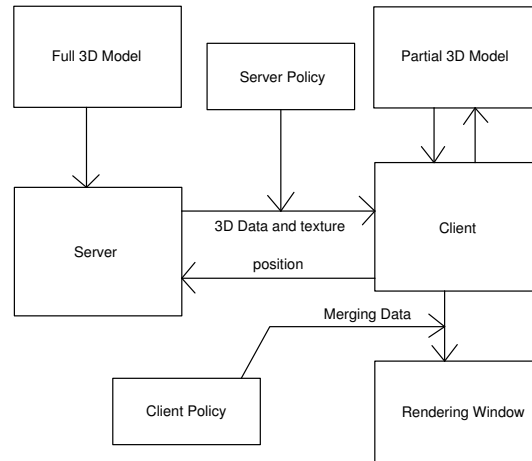


Fig. 11.1 The client-server paradigm.

The characteristics of system RV-RD are shown in Fig. 11.1. A server and a client are connected via a bandlimited channel. The server stores the full model, with high quality texture. At client side, the user interactively determines the particular view of interest, by communicating his or her position to the server. The server sends progressively compressed 3D data and texture, according to the user position and available bandwidth. The server never sends all its data (this would just happen if the bandwidth was infinite), but only the part that, according to the server policy, best fitting the user's needs and the available bandwidth. The client stores the 3D data and texture it receives from the server, according to its memory capacity. The available 3D data and texture are combined according to the client policy and the rendering engine provides the requested views.

In the RV-RD paradigm the client does not need to know the server policy and the server does not need to know the client policy. The client tries to use the available data the best way it can. The user can navigate in the 3D environment even if the connection between client and server is down, since the client will show only the data it has.

RV-RD avoids as much as possible the drawbacks of the two previously considered visualization paradigms and it may be worth noting that RV-CA and RV-I are special cases of RV-RD corresponding to different policies for client and server. If the server ignores the position of the client and sends all the model at the first request from the client, one obtains RV-CA where

the full 3D model is transmitted at once. If the server does not send any 3D data, but just a sequence of rendered images according to the user's position, one obtains RV-I.

This chapter essentially describes the implementation of a smart client for system RV-RD which is the basic ingredient of a RV-RD system and the most critical for the rendering results. The next section dwelves into the difficulties of efficient RV-RD policies in light of the variety of user's behaviours and it establishes a conceptual connection between the plenoptic function and such policies. Section 3 introduces basic affine warping notation and concepts used at the client in order to render a view, under the assumption that the client, besides geometry, has received a single picture. The rendering is obtained by a suitable affine transformation approximating the actual projective transformation on each triangle of the mesh. Section 4 describes how to include warping within discrete wavelet transform (DWT) analysis and synthesis in order to exploit the availability of N transmitted views at a client side for rendering purposes. Section 5 defines the client policy which accounts for the distortion due to the limited quality of the texture and geometry data available at a client side. Section 6 describes a technique for the real-time computation of the distortion defined in Section 5. Section 7 presents some experimental results showing the practical effectiveness of the client policy. Section 8 draws the conclusions.

11.2 The basic RV-RD framework

In the RV-RD system we envisage a server and a client, connected via a bandlimited channel. Characteristics of the server, of the client and of the network are known. However, finding a good policy for the server and the client is not trivial. The main problem consists in the ignorance, both by the server and the client, of what the user interaction with the world will be like. In order to appreciate this, consider that the user can be expected to navigate between a variety of different views; however, we do not know beforehand which views will be of interest. We also do not know in advance how much time (transmission resources) the user will choose to devote to any particular view. At one extreme the user may keep focused on a single view for a considerable period of time, waiting until very high quality imagery has been recovered before moving on. In this case, the interactive retrieval problem is tantamount to that of interactive image browsing, which is most elegantly addressed by progressive transmission of a single

scalably compressed image, formed at the server. One way to achieve this is to combine the JPIP interactive imaging protocol with a JPEG2000 compressed representation of the desired view [D.Taubman and R.Prandolini (2003)]. At the opposite extreme, the interactive user may select many different views in rapid succession, with the aim of understanding the scene geometry. This phase might itself be a precursor to later detailed inspection of some particular view of interest. All the intermediate cases between these two extremes should be considered during planning of the server and client policies. Close and accurate navigation in a 3D model should be allowed as well as an overall inspection. A slow navigation should be considered, as well as fast movements of the user's position.

In order to cope with all the possibilities, we assume that in a RV-RD system the server stores two types of data, forming the complete 3D environment:

- (1) a collection of views of the scene $V_{original}^i$;
- (2) the scene surface geometry, $G_{original}$.

As depicted in Fig. 11.2, the server delivers incremental contributions of these two types of data in the form of:

- (1) scalably compressed images V^i of the scene, computed from the collection of views $V_{original}^i$;
- (2) scalably compressed representation of the scene surface geometry G , obtained from the complete geometry $G_{original}$.

This provision is especially appropriate in view of the fact that the 3D environment is usually built from a collection of 3D views, merged together into a complete 3D model, and a collection of 2D images, which represent the texture of the environment. The 3D views may be obtained from the collection of 2D images or by other means. It is important to observe that while merging the 3D views usually leads to a more compact, complete and precise representation of reality, merging all the images taken from the scene might lead to loss of information. In fact, if the user looks at the 3D environment from a position where a photograph has been taken, sometimes the best choice might consist in serving the client directly with the scalably compressed representation of that photograph. If all the views were merged together, this could be not possible anymore. In Fig. 11.2, this means that the rendered view $V_{rendered}$ is identical to the corresponding image V^j .

Interestingly, though, this might not always be the best policy. If the

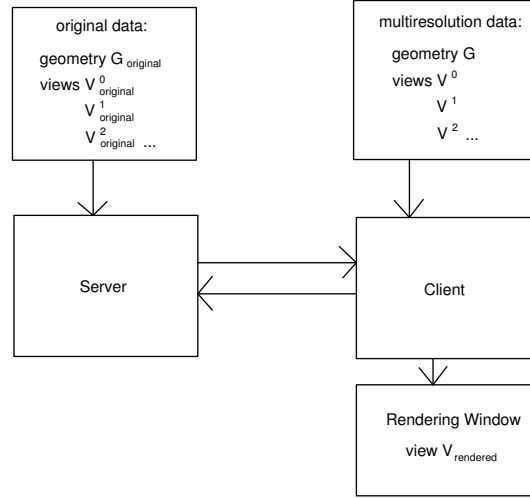


Fig. 11.2 Original and compressed data.

client has already received sufficient elements with sufficient quality from one or more nearby compressed view images, V^1 , V^2 , ..., it may be more efficient to send only the amount of geometric information required by the client to synthesize the requested view. In this case, the resulting bandwidth savings could be used to further augment the quality of these nearby original view images. It follows that even if the server has a large number of original view images, an efficient service policy would effectively sub-sample them based on the interactive user's navigation patterns. More generally, the server may choose to send some elements from V^i , while expecting the client to derive other aspects of the view from the previously delivered, but less closely aligned, original view images V^n .

It is worth pointing out that the compressed server images V^i do not necessarily correspond to the new views rendered by the client.

With respect to geometry, there exist a vast amount of work about progressive compression of 3D (non textured) meshes [Garland and Heckbert (1997)] [Rusinkiewicz and Levoy (2000)] [Hoppe (1996)] [Guskov *et al.* (1999)] [Schroeder *et al.* (1992)] and there are many ways to implement this concept.

The server of RV-RD does not generate new views or compress differential imagery. Instead, within an optimal policy, it determines and sends appropriate elements from a fixed set of scalably compressed bit-streams, so as

to provide its clients with the most appropriate data to render their desired views from. The framework RV-RD may thus be interpreted as fostering a greedy strategy for non-linear approximation of the plenoptic function [Zanuttigh *et al.* (2005)]. We will explain this concept. The server, with geometry G and views V^i , stores a representation of the plenoptic function. The plenoptic function must be approximated in order to be sent through the network and this is done through a greedy strategy, which means that the best immediate, or local, solution must be found. We would expect that a good strategy considers both view sub-sampling and rate-distortion criteria. The fact that efficient service policies can be expected to automatically sub-sample the existing content, contrasts the proposed approach with the previously mentioned predictive approach RV-I, where imagery is delivered for every view requested by the user.

A fundamental question an implementation of RV-RD system must reply is how the client should combine information from available original view images into a new view of interest, using the available description of the surface geometry. Such a possibility is instrumental to reuse the transmitted information and to avoid the waste of resources intrinsic of predictive system RV-I. The rest of this chapter describes a possible solution.

11.3 Rendering from a single view by affine warping

This section describes how the client can combine information from available 3D and texture data into a new view of interest. Here we assume that the client has already received the surface geometry G and one single image V^i from the server at a certain resolution, according to the server policy. We assume a triangular mesh representation for surface geometry G .

Wanted view V^* corresponds to the user's request. View V^* is rendered by a projection matrix P^* . Namely, matrix P^* projects nodes \mathbf{X}_i of G from the 3D space onto the 2D points \mathbf{J}_i , of the domain of V^* , as shown in Fig. 11.3, according to the following relationship in homogeneous coordinates [Hartley and Zisserman (2000)]:

$$\begin{bmatrix} w\mathbf{J}_i \\ w \end{bmatrix} = P^* \begin{bmatrix} \mathbf{X}_i \\ 1 \end{bmatrix} \quad (11.1)$$

The structure of P^* is:

$$P^* = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R^T \ t] \quad (11.2)$$

where R and t are the rotation matrix and translation vector of the user reference system, with respect to the world reference system. Entries f_x , f_y , c_x , c_y and s are the intrinsic parameters of the virtual camera used by the client in order to render the world to the user.

We assume that V^i was taken by a pre-calibrated camera with projection matrix P^i . Let Δ_k^* and Δ_k^i denote the triangular patches obtained by projecting the same triangle Δ_k of 3D mesh G on V^* by P^* and on V^i by P^i (Fig. 11.3). Let us denote as $V^*(\Delta_k^*)$ and as $V^i(\Delta_k^i)$ the portions of image V^* and V^i supported by Δ_k^* and Δ_k^i respectively.

$$(V^i(\Delta))[\mathbf{n}] = \begin{cases} V^i[\mathbf{n}] & \mathbf{n} \in \Delta \\ 0 & \mathbf{n} \notin \Delta \end{cases}$$

where $\mathbf{n} = [n_1, n_2]$ is a sample in the image.

Of course some of the projected triangles may be hidden in one image, but not in the other. If Δ_k^* is hidden, then Δ_k^i is not involved in rendering, while if Δ_k^i is hidden, Δ_k^* is a “hole” in V^* , i.e. an area which cannot be rendered from V^i .

Apart from the holes, each exposed triangle Δ_k^* is rendered by an affine warping: \mathcal{T}_k^i is the mapping on image V^i which takes Δ_k^i to Δ_k^* . The resulting portion of image is $(\mathcal{T}_k^i V^i)(\Delta_k^*)$.

The wanted view V^* is rendered as the sum of all the triangular image patches produced by operators \mathcal{T}_k^i delivering the result of each triangular patch Δ_k^* . We denote as \mathcal{T}^i such a global operator, i.e.:

$$V^* = \mathcal{T}^i V^i = \sum_k (\mathcal{T}_k^i V^i)(\Delta_k^*) \quad (11.3)$$

Note that while the triangle vertices are correctly transformed with respect to projective transformation P^* by the affine warping, such a transformation only approximates the action of P^* in their interior, which is an homography between triangles. However, this error can be rendered arbitrarily small by reducing the size of the surface mesh elements. Affine warping is preferable to projective warping since it is supported by most graphic engines and it is faster. Algorithms for fast warping of images are well known in computer graphics [Foley *et al.* (1995)].

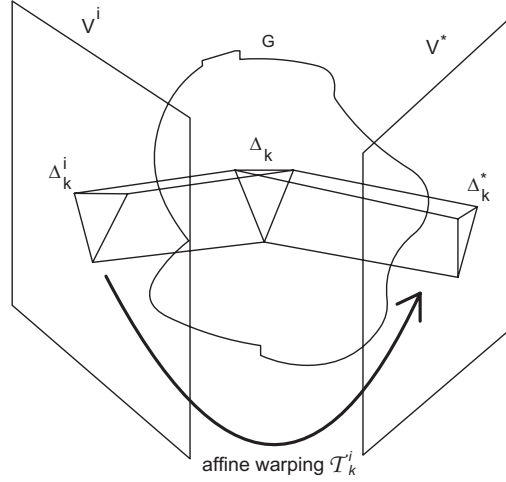


Fig. 11.3 Rendering V^* from geometry G and view V_i .

11.4 Rendering from multiple views by DWT analysis, warping and DWT synthesis

In the general case, the client has available many images $V^i, i = 1 \dots N$ previously sent by the server. Assume that V^i were taken by pre-calibrated cameras, each one with projection matrix $P^i, i = 1 \dots N$. One way to combine information from multiple original view images, V^0, V^1, \dots , is to simply average the results obtained by mapping each of them onto the desired view, i.e.,

$$V^* = \frac{1}{N} \sum_{i=0}^{N-1} \mathcal{T}^i V^i \quad (11.4)$$

Unfortunately, any imperfection in the surface geometry description produces misalignment amongst the separate renderings $\mathcal{T}^i V^i$, so that averaging tends to blur high frequency spatial features. Also, the simple average shows no preference for one possible rendering over another. Indeed some view may be less noisy or closer than the others to the wanted view, therefore it would be sensible to give a higher weight to its contribution. Fig. 11.4(a) and Fig. 11.4(b) show two objects, a Santa Claus and a log. Fig. 11.5(a) and Fig. 11.5(b) show the effects of rendering a view by averaging two images ($N = 2$) for the Santa Claus and for the log.



Fig. 11.4 The objects: a) Santa Claus; b) The log.



Fig. 11.5 Rendering by averaging: a) Santa Claus; b) The log.

An alternative strategy is to select a single most appropriate original view image from which to render each triangle. We refer to this as stitching: in $(\mathcal{T}_k^{i_k} V^{i_k, k})(\Delta_k^*)$, i_k is the “best stitching source” for the k^{th} triangle. The rendered image becomes:

$$V^* = \sum_k (\mathcal{T}_k^{i_k} V^{i_k})(\Delta_k^*) \quad (11.5)$$

The problem, of course, is identifying the best stitching source for each Δ_k^* . Although stitching avoids the blurring problem, it tends to produce visible discontinuities at the boundaries between adjacent triangles which are rendered from different original source views. This is because the surface geometry will inevitably contain modeling errors, and the considered rendering process does not account for illumination-dependent shading effects. This artifact due to stitching can be seen in Fig. 11.6, and it is rather evident on the leg of the model.



Fig. 11.6 Rendering from two views by stitching not in the DWT domain.

To hide these artifacts, and also to open the door to resolution-dependent stitching algorithms, we prefer to form V^* in the discrete wavelet transform or DWT domain [Taubman and Marcellin (2002)]. The process is shown in Fig. 11.7. Each possible rendering is generated through warping of each image V^i , which gives $(\mathcal{T}^i V^i)$, $i = 1 \dots N$. This process corresponds to the one described in Section 11.3. Each possible rendering is subjected to DWT analysis. This produces a collection of subbands: we call $\mathbf{b} \equiv (\theta, d)$ a generic subband, where $\theta \in \{0, 1, 2, 3\}$ stands for $\{LL, HL, LH, HH\}$ and $d \in \{1, 2, \dots, D\}$ is the level of wavelet decomposition, where D is the number of DWT levels (“depth” in the tree), see Fig. 11.8.

Stitching is carried out within the individual subbands to produce subbands \mathbf{b} of V^* , as shown in Fig. 11.7. Each triangular portion $V_{\mathbf{b}}^*(\Delta_k^*)$ of each subband \mathbf{b} of the destination image V^* is chosen by a suitable policy among the DWT decompositions of $(\mathcal{T}_k^i V^i)$, $i = 1 \dots N$ relative to the same subband.

At the end, V^* is recovered by DWT synthesis. Experimental evidence for the benefits of the DWT-based stitching is provided in Section 11.7. The policy used for the stitching strategy is explained in the next section.

In order to formalize the procedure for rendering from multiple views let us introduce some notation.

Let $\mathcal{A}_{\mathbf{b}}$ denote the subband analysis operator: it is a sequence of filtering and downsampling operators [Taubman and Marcellin (2002)], which transform an image $V[\mathbf{n}]$ in the matrix of coefficients $V_{\mathbf{b}}[\mathbf{n}]$ at subband \mathbf{b} , as shown in Fig. 11.8(a):

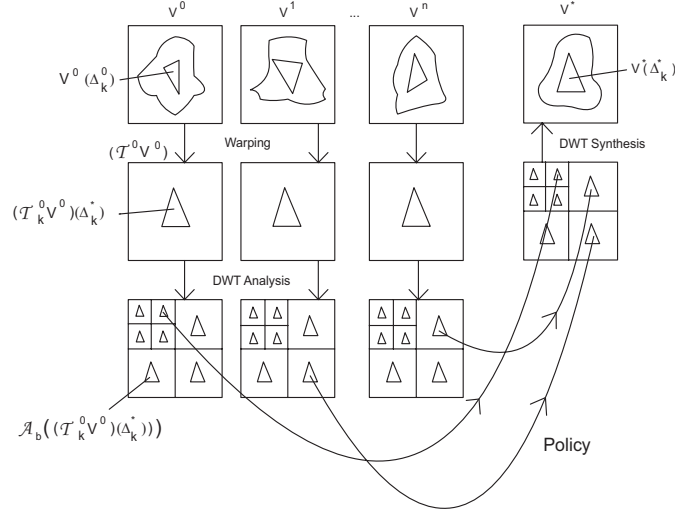


Fig. 11.7 Stitching in DWT domain.

$$V_{\mathbf{b}}^i = \mathcal{A}_{\mathbf{b}}(V^i) \equiv V_{\mathbf{b}}^i[\mathbf{n}] = (\mathcal{A}_{\mathbf{b}}(V^i))[\mathbf{n}]$$

Let $\mathbf{S}_{\mathbf{b}}$ denote the subband synthesis operator: it is a sequence of up-sampling and filtering operators, which brings the contribution of a matrix of wavelet coefficients $V_{\mathbf{b}}[\mathbf{n}]$ at subband \mathbf{b} to image $V[\mathbf{n}]$, as shown in Fig. 11.8(b):

$$V^i = \sum_{\mathbf{b}} \mathcal{S}_{\mathbf{b}}(V_{\mathbf{b}}^i) \quad (11.6)$$

Image V^* is obtained through synthesis from all the coefficients $V_{\mathbf{b}}^*$ at subbands \mathbf{b} :

$$V^* = \sum_{\mathbf{b}} \mathcal{S}_{\mathbf{b}}(V_{\mathbf{b}}^*) \quad (11.7)$$

The coefficients of each subband can also be seen as sum of the coefficients with support Δ_k^* at subbands \mathbf{b} , which can be written as:

$$V^* = \sum_{\mathbf{b}} \mathcal{S}_{\mathbf{b}} \left(\sum_k (V_{\mathbf{b}}^*(\Delta_k^*)) \right) \quad (11.8)$$

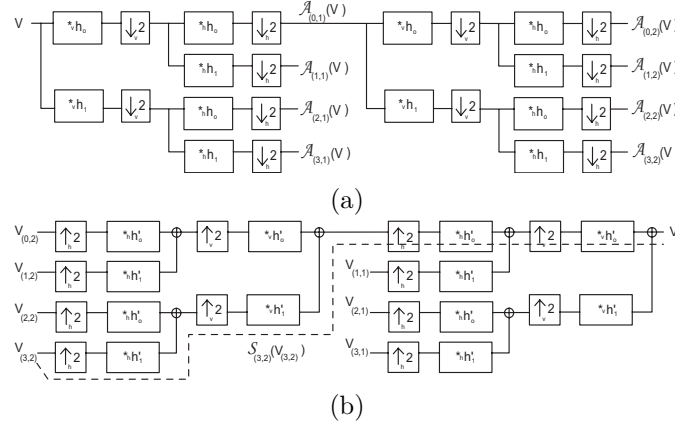


Fig. 11.8 a) analysis operators $\mathcal{A}_{\mathbf{b}}$; b) synthesis operators $\mathbf{S}_{\mathbf{b}}$ (the contribution of $\mathbf{b} = (3,2)$ is shown as a dotted line).

11.5 The client policy

In expression (11.8) each portion of coefficients $V_{\mathbf{b}}^*(\Delta_k^*)$ in subband \mathbf{b} comes from one of the warped triangle coefficients $(\mathcal{T}_k^i V^i)(\Delta_k^*)$, $i = 1 \dots N$, in the subband \mathbf{b} , i.e.,

$$V_{\mathbf{b}}^*(\Delta_k^*) = [(\mathcal{A}_{\mathbf{b}} \circ \mathcal{T}^{i_{\mathbf{b},k}})(V^{i_{\mathbf{b},k}})](\Delta_k^*) \quad (11.9)$$

where $i_{\mathbf{b},k}$ identifies the selected source image, $V^{i_{\mathbf{b},k}}$, for triangle Δ_k in subband \mathbf{b} . We call this the stitching source. We must determine a policy for choosing the “best” stitching source $i_{\mathbf{b},k}$. Such a source depends on the subband \mathbf{b} and the triangle index k .

When selecting one source instead of another, the client suffers a certain error bringing various types of distortion in the rendered views which must be minimized. The strategy for defining policy function $i_{\mathbf{b},k}$ is the minimization of such an error, which will be formally expressed in this section.

There are three sources of error. The first source of error, which will be called quantization error for simplicity, is the quality of images that are sent by the server: the server usually sends data in a progressive fashion (as JPIP) and at a certain time the client owns images of a certain quality, usually different from the full quality of the pictures available at the server side.

A second error source is the warping process, which is bound to introduce some distortion. Such a distortion is correlated with the viewing geometry, indeed a patch of the model which is seen well from the user could be present in the original images from a very acute angle. This means that warping will expand the patch in a relevant way and high frequencies could be missing. The distortion must combine affine approximation and viewing geometry. For example, let us assume that two photos have been sent by the server to the client: the first image has a very high quality and the second one a low quality. In the same geometric conditions, the best quality picture will be used to create the final picture, in the way described in previous section. However, it could happen that a certain patch, seen by the user, is best described, in low and high frequencies, by a lower quality picture which is more parallel to the viewer.

The third source of error is due to geometric modeling: the geometry G_{server} available at the server side is not exactly identical to the real scene. Errors in calibration, acquisition and reconstruction may be present. Furthermore the geometry G_{client} available at the client side in general is a simplified or compressed version of the geometry G_{server} , and this introduces a quantization error.

We assume that the original view images were compressed in the DWT domain (e.g., using JPEG2000). The full quality images at the server side and their compressed versions (sent by the server), represented by Expression 11.6, available at client side are different because of quantization. This difference is an error image e^i . We call \mathbf{b}' a subband of V^i , and \mathbf{b} a subband of the destination image V^* , made by warped contributions. We call B the complete set of subbands \mathbf{b} , and B' the complete set of subband \mathbf{b}' . A quantization error is associated to each matrix of coefficients $V_{\mathbf{b}'}^i(\Delta_k^i)$ in the subband \mathbf{b}' of V^i falling within the scope of triangle Δ_k^i and is called $e_{\mathbf{b}',k}^i(\Delta_k^i)$.

In order to understand how the quantization error intrinsic to the limited quality images (11.6) affects the final rendered view V^* , consider the error propagation chain due to the DWT with the aid of Fig. 11.9.

At client side, because of the warping and of the subsequent analysis, the limited image quality error propagates down to the warped triangles $(\mathcal{T}_k^i V^i)(\Delta_k^*)$ in all subbands. Such triangles are selected as the coefficients of 11.9 of V^* at subband \mathbf{b} , V^* is obtained through synthesis (11.8) and it is affected by the errors as shown in Fig. 11.9. Once the stitching source $i_{\mathbf{b},k}$ has been selected, we know that error $e_{\mathbf{b}',k}^{i_{\mathbf{b},k}}(\Delta_k^{i_{\mathbf{b},k}})$, due each to subband \mathbf{b}' , affects image $V^{i_{\mathbf{b},k}}$ through synthesis, which gives the error:

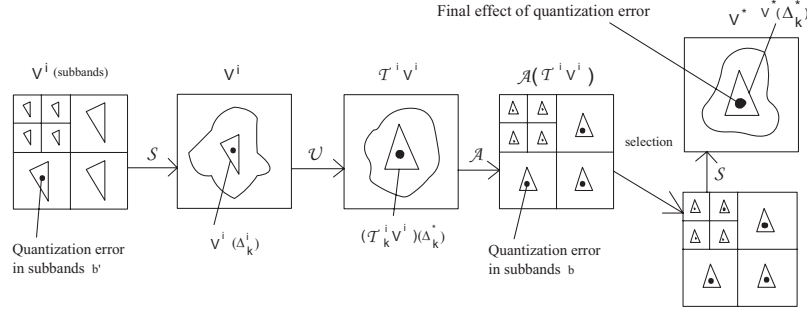


Fig. 11.9 Quantization error.

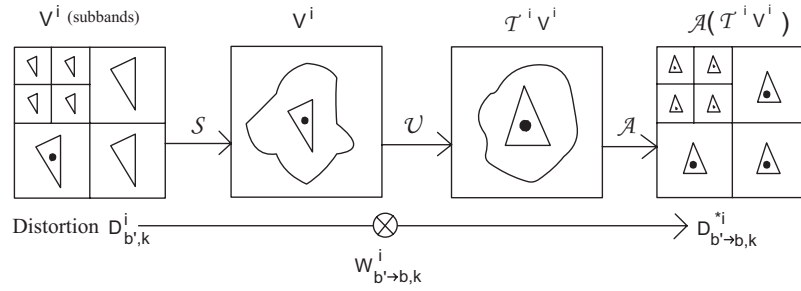


Fig. 11.10 Propagation of the quantization error.

$$e^{i_{b,k}} = \sum_{b' \in B'} \mathcal{S}_{b'} \left(\sum_k e_{b',k}^{i_{b,k}} (\Delta_k^{i_{b,k}}) \right) \quad (11.10)$$

Error $e^{i_{b,k}}$ propagates in the subband b of the warped image, as shown in Fig. 11.10, and brings to V^* the error:

$$e_b^* = [(\mathcal{A}_b \circ \mathcal{T}^{i_{b,k}}) (e_{b,k}^{i_{b,k}})] (\Delta_k^*) \quad (11.11)$$

If we analyze the contribution of only one source triangle Δ_k^i and only one source subband b' on a destination subband b , we can write:

$$e_{b' \rightarrow b,k}^{*i_{b,k}} = [(\mathcal{A}_b \circ \mathcal{T}^{i_{b,k}}) (\mathcal{S}_{b'} e_{b',k}^{i_{b,k}})] (\Delta_k^*) \quad (11.12)$$

A useful parameter to evaluate the error propagation is the mean square error of the error image, which we call distortion. The distortion of error $e_{\mathbf{b}'}^{i_{\mathbf{b},k}}(\Delta_k^{i_{\mathbf{b},k}})$ is:

$$D_{\mathbf{b}',k}^{i_{\mathbf{b},k}} = \|e_{\mathbf{b}'}^{*i_{\mathbf{b},k}}(\Delta_k^{i_{\mathbf{b},k}})\|^2 \quad (11.13)$$

The correlated distortion at the \mathbf{b} subband in V^* is:

$$D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}} = \|e_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}}\|^2 \quad (11.14)$$

Parameter $D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}}$ only accounts for the error contribution from the \mathbf{b}' subband of $V^{i_{\mathbf{b},k}}$ to the \mathbf{b} subband of V^* .

The distortion at the \mathbf{b} subband of V^* due to the limited quality of the images available at client side, warping and DWT analysis and synthesis can be approximated as:

$$D_{\mathbf{b},k}^* = \sum_{\mathbf{b}' \in B'} D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}} \quad (11.15)$$

The distortion of one triangle is:

$$D_k^* = \sum_{\mathbf{b} \in B} D_{\mathbf{b},k}^* \quad (11.16)$$

The complete distortion of the final image V^* is:

$$D^* = \sum_k D_k^* \quad (11.17)$$

The best stitching source $i_{\mathbf{b},k}$ for each triangle Δ_k is the one for which $D_{\mathbf{b},k}^*$ is smallest. Note that the best stitching source could potentially differ from subband to subband. We can now write the policy function as:

$$i_{\mathbf{b},k} = \operatorname{argmin}(D_{\mathbf{b},k}^*) \quad (11.18)$$

11.6 How to compute the distortion contribution ?

In an environment where the user must be able to navigate in real-time it is not possible for each error associated to each triangle Δ_k and for each source V_i , to compute synthesis, warping, analysis and to get the terms

of distortion (11.30) in real-time. Moreover, the client does not know the original images and the actual geometry in order to compute the difference between them and the images it has available. Such differences are indeed available only at the server side. At the client one may estimate the needed information and compute the differences on the basis of such estimates. The estimation methods of [Taubman and Secker (2003)] can be used in this task. This section will describe a practical method for computing distortion (11.30) in real-time. For simplicity, let us begin with the computation of just the quantization error (11.15), ignoring the second and third term of (11.30).

Basic distortion term (11.14) can be evaluated as the product of the previously introduced quantization errors $D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{i_{\mathbf{b},k}}$ of 11.13 by suitable weights which can be stored in a look-up table (see Fig. 11.10), namely,

$$D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}} = W_{\mathbf{b}' \rightarrow \mathbf{b},k}^{i_{\mathbf{b},k}} D_{\mathbf{b}',k}^{i_{\mathbf{b},k}} \quad (11.19)$$

This formula is correct if we assume uncorrelated quantization errors, or orthogonal basis vectors, as is the case of the DWT basis vectors. This formula is an approximation, because we assign all of the incurred distortion to Δ_k^i ignoring the fact that the error signal is smeared out by the overlapping DWT basis vectors.

Hence, from (11.19) Equation (11.15) can be rewritten as:

$$D_{\mathbf{b},k}^* = \sum_{\mathbf{b}' \in B'} W_{\mathbf{b}' \rightarrow \mathbf{b},k}^{i_{\mathbf{b},k}} D_{\mathbf{b}',k}^{i_{\mathbf{b},k}} \quad (11.20)$$

By comparing (11.20) with (11.15) one notices that $W_{\mathbf{b}' \rightarrow \mathbf{b},k}^{i_{\mathbf{b},k}}$ must account for the effects of synthesis, warping and analysis.

11.6.1 *Effects of the affine warping*

Let us first describe how to consider the effects of the affine warping. An affine transformation is a composition of translation, rotation, scaling and shear. The actual results of analysis and synthesis slightly depend on the position of the error in the image. Since we are only interested in an approximation, in this context the translation contribution can be ignored.

Therefore, we will consider affine transformations mapping (x_1, y_1) to (x_2, y_2) according to the following expression:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad (11.21)$$

where α is the angle of the 2D rotation; s is the shear factor and c_1 and c_2 the scaling factors.

First, we quantize the possible affine transformations, in order to have a small number of possible warping operators. Angle α can be linearly quantized as:

$$\alpha_i = 2\pi * i/N_\alpha, \text{ with } i = 0 \dots N_\alpha - 1$$

Shear s usually is very close to zero because the same triangle, from different points of view is rotated and scaled but not stretched, since the parameters of the user's camera do not change. Shear s is quantized in a cubic way as:

$$s = K_s * (2i/N_s - 1)^3, \text{ with } K_s \text{ constant value and } i = 0 \dots N_s$$

Scaling factors c_1 and c_2 are frequently equal to one, since usually most surface triangles are parallel to the viewer. Therefore, it makes sense to sample c_1 and c_2 more densely near one.

$$c_1 = K_1 * (2i_1/N_1 - 1)^3 + 1, \text{ with } K_1 \text{ constant value and } i_1 = 0 \dots N_1$$

$$c_2 = K_2 * (2i_2/N_2 - 1)^3 + 1, \text{ with } K_2 \text{ constant value and } i_2 = 0 \dots N_2$$

Values $W_{\mathbf{b}' \rightarrow \mathbf{b}, k}^{i_{\mathbf{b}', k}}$ are computed for each possible quantization of the affine transformation \mathcal{T}_k^i . The quantized values can be denoted as $W(\mathbf{b}, \mathbf{b}', \alpha, s, c_1, c_2)$ for each possible source subband \mathbf{b}' , destination subband \mathbf{b} and affine warping parameters.

The size of the look-up table depends on the above parameters N_α , N_s , N_1 , N_2 and the number of subbands, as explained in the next section. It was experimentally verified that with $N_\alpha = N_s = N_1 = N_2 = 10$, and with $D = 6$ subband levels, the approximation is quite good and the table is not too large (around 30 Mbytes).

11.6.2 Effects of analysis and synthesis

In order to evaluate the effects of analysis and synthesis together with warping, consider an impulse $e_{\mathbf{b}', k}^i$, i.e., a difference image with a value 1 placed at the center and compute synthesis $\mathcal{S}_{\mathbf{b}'}$, warping with s, c_1, c_2 and analysis $\mathcal{A}_{\mathbf{b}}$ in order to get $W(\mathbf{b}, \mathbf{b}', \alpha, s, c_1, c_2)$.

Fig. 11.11(a) shows a 16x16 example of $e_{\mathbf{b}', k}^i$ with a single sample at the center of subband $\mathbf{b}' = (0, 4)$. After synthesis a blob appears in the form of the 256x256 image of Fig. 11.11(b). Fig. 11.11(c) shows the result of warping by an affine transformation. The result of the analysis in the subband $\mathbf{b} = (0, 4)$, the squared value of which gives $W(\mathbf{b}, \mathbf{b}', \alpha, s, c_1, c_2)$, is shown by the 16x16 image of Fig. 11.11(d).

There is a point worth noting. The position of the sample affects the

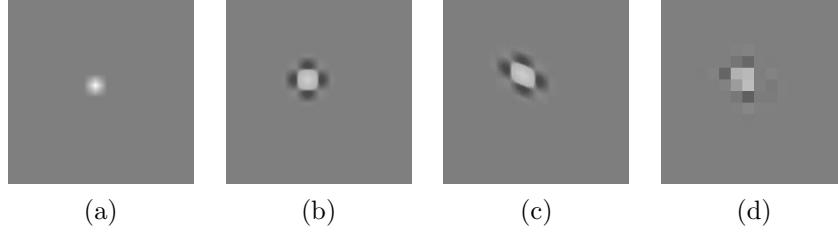


Fig. 11.11 Computation of $D_{jd \rightarrow hm, \alpha, s, c_1, c_2}$: (a) impulsive error in subband $d = 4, j = 1$ (b) result of synthesis (c) result of warping (d) result of analysis in subband $h = 4, m = 1$.

result of analysis and synthesis in DWT. An impulse error placed at the center of the subband approximates only an error at the origin, but errors can be positioned anywhere in the original image. More sensible values of W are obtained if one places the impulse errors in a certain number of positions around the center and then averages the results.

This procedure gives satisfactory results because analysis and synthesis are linear operators and error image $e_{\mathbf{b}',k}^i$ is the sum of scaled impulse contributions. Experiments showed that the weights $W(\mathbf{b}, \mathbf{b}', \alpha, s, c_1, c_2)$ obtained by this procedure approximate very well the actual values of $D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i\mathbf{b},k}$ given by (11.19).

11.6.3 Effects of extra-band energy

Operator \mathcal{T}_k^i stretches the portion of image V^i within triangle Δ_k^i by ratio $|\Delta_k^*| / |\Delta_k^i|$ between the areas of warped and original triangles. ($|\Delta_k|$ denotes the area of triangle Δ_k) and amplifies the image energy roughly by the same amount. Assuming an orthonormal transform, we can say that given unitary distortion $D_{\mathbf{b}',k}^i = \|e_{\mathbf{b}',k}^i\|^2 = 1$, the contribution of the error from subband \mathbf{b}' to the warped image $\mathcal{T}_k^i V^i$ is:

$$\|\mathcal{T}_k^i \mathcal{S}_{\mathbf{b}'}(e_{\mathbf{b}',k}^i)\|^2 = W_{\mathbf{b}' \rightarrow \mathbf{b},k}^i = |\Delta_k^*| / |\Delta_k^i| \quad (11.22)$$

The energy is preserved in all the subbands by DWT analysis. Incidentally, the 9/7 biorthogonal wavelet transform used for our experiments is very nearly orthonormal, subject to appropriate normalization of the subband samples. Therefore, if $(\mathcal{T}_k^i V^i)$ is decomposed in subbands the total energy in each subband stays the same, and,

$$\sum_{\mathbf{b}'} W_{\mathbf{b}' \rightarrow \mathbf{b},k}^i = \|\mathcal{T}_k^{\mathbf{b},k}(\mathcal{S}_{\mathbf{b}'}(e_{\mathbf{b}',k}^i))\|^2 = |\Delta_k^*| / |\Delta_k^i| \quad (11.23)$$

From (11.19) we see that distortion $D_{\mathbf{b}',k}^i$ is scaled by $W_{\mathbf{b}' \rightarrow \mathbf{b},k}^i$ in order to obtain $D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i}$, hence:

$$\begin{aligned} D_k^* &= \sum_{\mathbf{b}' \in B'; \mathbf{b} \in B} D_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i} = \\ &= \sum_{\mathbf{b}' \in B'; \mathbf{b} \in B} (W_{\mathbf{b}' \rightarrow \mathbf{b},k}^i D_{\mathbf{b}',k}^i) = \\ &= \sum_{\mathbf{b}' \in B'} D_{\mathbf{b}',k}^i \sum_{\mathbf{b} \in B} W_{\mathbf{b}' \rightarrow \mathbf{b},k}^i = \\ &= |\Delta_k^*| / |\Delta_k^i| \sum_{\mathbf{b}' \in B'} D_{\mathbf{b}',k}^i \end{aligned} \quad (11.24)$$

The above expression apparently suggests that the total distortion in the warped triangle (left hand side) is roughly independent from the affine operator \mathcal{T}_n^i , since the total distortion in the source triangles $\sum_{\mathbf{b} \in B} D_{\mathbf{b},k}^i$ should be nearly proportional to their areas $|\Delta_k^i|$. However, this is not the case for two reasons. Firstly, \mathcal{T}_k^i must be a bandlimited warping operator, so that $\|\mathcal{T}_k^i(\mathcal{S}_{\mathbf{b}'}(e_{\mathbf{b}',k}^i))\|^2 = |\Delta_k^*| / |\Delta_k^i| \cdot F \left(\|\mathcal{T}_k^i(\mathcal{S}_{\mathbf{b}'}(e_{\mathbf{b}',k}^i))\|^2 \right)$, where $F \left(\|\mathcal{T}_k^i(\mathcal{S}_{\mathbf{b}'}(e_{\mathbf{b}',k}^i))\|^2 \right)$ is the fraction of the energy of $\|\mathcal{T}_k^i(\mathcal{S}_{\mathbf{b}'}(e_{\mathbf{b}',k}^i))\|^2$ within the Nyquist frequency. Secondly, expansive operators \mathcal{T}_k^i cannot predict the highest frequency details of V^* at all. Both effects can be taken into account by extending the sums in equation (11.20) to include subbands from a set of hypothetical resolutions above those of the original images.

Fig. 11.12(a) indicates that if an affine transformation shrinks a triangle, most of the energy comes from the lower subbands. Usually the energy from the higher level subbands of V^i is lost, since their contribution does not appear in the final image V^* . The picture is different in Fig. 11.12(b) showing an affine transformation expanding a triangle. It can be seen that the energy which should be transferred to higher levels of V^* is missing, since the corresponding subbands of V^i do not exist. This must be taken into account when computing the total distortion of Equation (11.20) together with the dual fact of the energy going to non-existing subbands in case of shrinking, since it does not appear in the final rendered view V^* .

This problem can be solved adding extra-band energy in the source image: the set of subbands of V^i becomes $B'' = B' \cup \mathbf{b}_{EB}$ with an extra

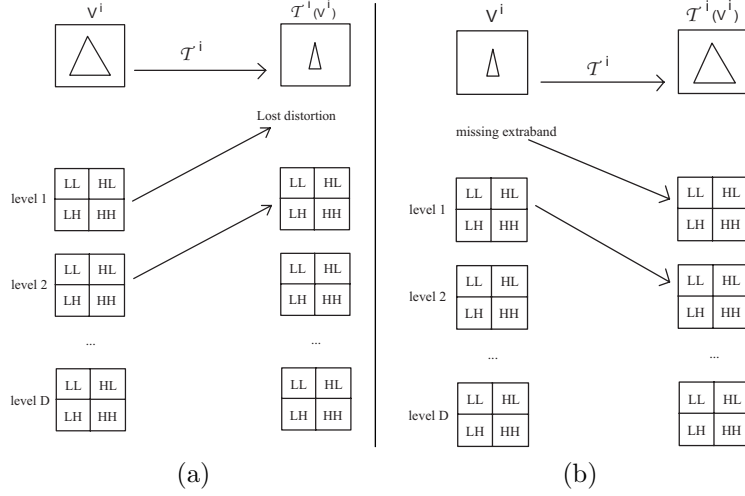


Fig. 11.12 Effects of (a) Shrinking (b) Expansion

level \mathbf{b}_{EB} added. In this extra-band we have the source distortions $D_{\mathbf{b}_{EB},k}^i$, which we estimate by projecting each source image V^i onto the other in turn and taking the maximum of the energy produced by such projections. This in practice means that we consider the energy of the source image with more detail. Indeed, it is not necessary to know the real energy of the details which were present in the real 3D model at higher resolution, since they will never appear if no image with such details is available. However, if the details exist in one of the images, they must be taken into account.

Equation (11.20) with the extra-band contribution accounted for by \mathbf{b}_{EB} becomes:

$$D_{\mathbf{b},k}^* = \sum_{\mathbf{b}' \in B''} W_{\mathbf{b}' \rightarrow \mathbf{b},k}^{i_{\mathbf{b},k}} D_{\mathbf{b}',k}^{i_{\mathbf{b},k}} \quad (11.25)$$

The extra contribution on the right hand side of Equation (11.25), shown in Fig. 11.12(b), grows with $|(\Delta_k^*)| / |\Delta_k^{i_{\mathbf{b},k}}|$. Instead, the target distortions of hypothetical subbands, as shown in Fig. 11.12(a), represent an unconsidered contribution to the left hand side of equation (11.25), which grows with $|\Delta_k^{i_{\mathbf{b},k}}| / |(\Delta_k^*)|$. As a result, we expect to find that the best stitching source $i_{\mathbf{b},k}$ is the one for which $\mathcal{T}_k^{i_{\mathbf{b},k}}$ is most contractive, all other things being equal. Of course, this usually does not happen: at a certain time,

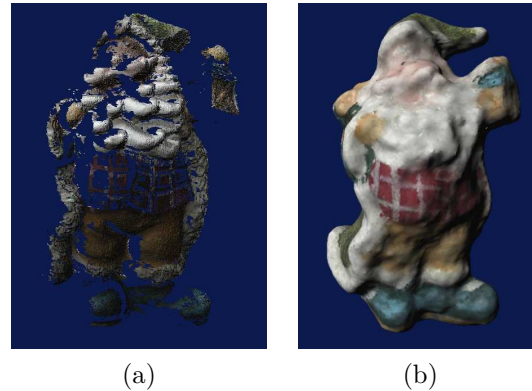


Fig. 11.13 Santa Claus from a 3D model obtained by: (a) range-camera (b) 3D passive reconstruction from a set of photographs

images can be present at different resolutions and this strongly affects the decision.

11.6.4 *Effects of geometry*

The error on V^* due to wrong or approximate geometry is considered in this section.

The reliability of the 3D geometry depends on different causes. First of all, the accuracy of the acquisition and 3D modeling tools affects reliability. For example, range-cameras can scan objects with a precision of a few tenths of a micron, while passive 3D reconstruction methods can be affected by errors of millimeters. Fig. 11.13 shows two rendered views of Santa Claus; the one in Fig. 11.13(a) comes from a 3D model obtained by a range-camera and is much more precise than the view in Fig. 11.13(b), coming from a 3D model obtained by a passive method based on a set of photographs (see Chapter 12).

A second source of error comes from quantization and transmission of multi-resolution 3D meshes from server to client: the model at the client side is not the same as the original 3D model available at the server. Furthermore, if geometry and texture are acquired by different instruments (e.g. a range camera for the former and a photocamera for the latter) possible calibration errors between the two instruments can produce misalignments between 3D data and texture.

All these sources contribute a geometric error: uncertainty in the surface

geometry translates into uncertainty in the parameters of the transformations \mathcal{T}_k^i , more specifically affected by a translational uncertainty, which has been studied in [Taubman and Secker (2003)]. Its effect may be modeled by augmenting each term $D_{\mathbf{b},k}^*$ in Equation (11.15) by a second contribution of the form:

$$D_{\mathbf{b},k}^* = \sum_{\mathbf{b}' \in B'} W_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}} \left(D_{\mathbf{b}',k}^{*i_{\mathbf{b},k}} + \left[g_k^{i_{\mathbf{b},k}} |\omega_{\mathbf{b}'}|^2 \right] E_{\mathbf{b}',k}^{i_{\mathbf{b},k}} (\Delta_k^{i_{\mathbf{b},k}}) \right) \quad (11.26)$$

which is function on the subband \mathbf{b}' from which the texture is taken, the triangle Δ_k^* and the view V^i . Most specifically g_k^i accounts for the influence on the geometry error of the actual shape of Δ_k^* over V^* ; $\omega_{\mathbf{b}'}$ is “representative” of the spatial frequencies belonging to subband \mathbf{b}' and expresses the influence of geometric error over such subband; $E_{\mathbf{b}',k}^{i_{\mathbf{b},k}} (\Delta_k^{i_{\mathbf{b},k}})$ is the energy of subband \mathbf{b}' inside triangle Δ_k^i .

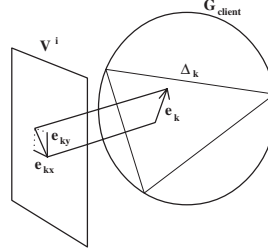
If we knew the power spectrum in the subband, i.e., how the energy density were distributed over the subband \mathbf{b}' , it would be logical to choose $\omega_{\mathbf{b}'}$, for example, as the maximum point of the centroid of the power spectrum. As we only know the energy of the subband (i.e., the integral of the power spectrum over the subband itself) we can simply take $\omega_{\mathbf{b}'}$ as the middle of the subband, according to:

$$\omega_{\mathbf{b}'} = \begin{cases} (\pi/2^d, \pi/2^d) & \text{when } \theta = 0 \\ ((\pi * 2)/2^d, \pi/2^d) & \text{when } \theta = 1 \\ (\pi/2^d, (\pi * 2)/2^d) & \text{when } \theta = 2 \\ ((\pi * 2)/2^d, (\pi * 2)/2^d) & \text{when } \theta = 3 \end{cases} \quad (11.27)$$

A more rigorous approach would fit a model to the power spectrum, and can be found in [Taubman and Secker (2003)].

For the computation of g_k^i one should consider how the wrong position of the mesh nodes (which are vertexes of the mesh triangles) of geometry G_{client} available at server side, i.e., an error in the 3D space, translates to an error on the 2D space of image V^* . This error corresponds to the error associated with each triangle Δ_k of G_{client} , and can be computed as the distance from the baricenter of Δ_k and the surface G_{server} (which is supposed to be the correct geometry of the object). That error is a vector \mathbf{e}_k . It can be projected over V^i and can be projected over V^i : it becomes $\mathbf{e}_k^i = (\mathbf{e}_{kx}^i, \mathbf{e}_{ky}^i)$, as shown in Fig. 11.14.

When a view V^* is rendered from V^i , the error of view V^i could be cancelled by the error of view V^* . For example, if $V^i \equiv V^*$, error in

Fig. 11.14 Geometric error \mathbf{e}_k on V^i .

reprojection of V^i cannot be distinguished since V^* is taken from the same point of view (see Fig. 11.15). Thus, error can be approximated as:

$$g_k^i = \sqrt{\|(\mathbf{e}_{kx}^i - \mathbf{e}_{kx}^*)\|^2 + \|(\mathbf{e}_{ky}^i - \mathbf{e}_{ky}^*)\|^2} \quad (11.28)$$

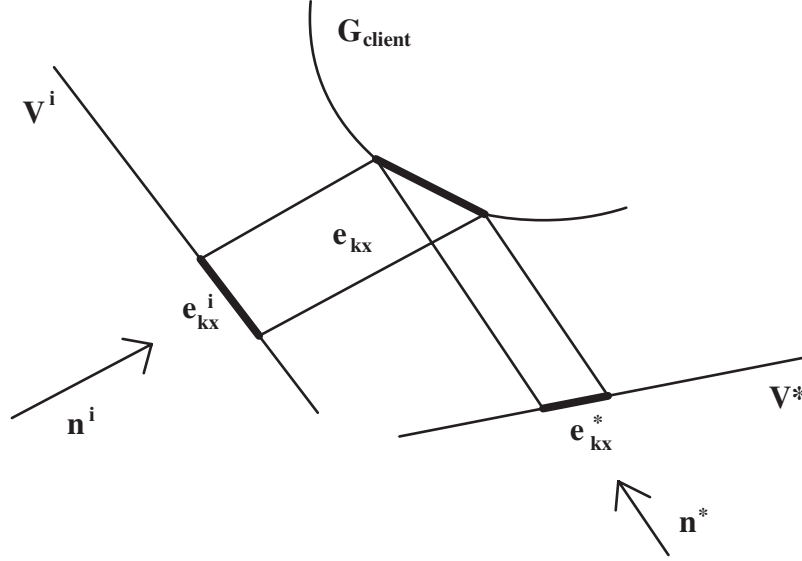
Error along the z axis (along the line of sight of the camera) does not appear here, since its contribution is included if it corresponds to an error along $(\mathbf{e}_{kx}^*, \mathbf{e}_{ky}^*)$.

This way of evaluating g_k^i suffers some drawbacks. First, it assumes that G_{server} is the real surface of the object, ignoring the fact that G_{server} may be affected by modeling errors. Also, it assumes that g_k^i is computed for each triangle and each new view V^* , which is a very time-consuming task. Furthermore, since g_k^i should be sent for each triangle (because the client does not know G_{server}), the network bit-rate is increased.

Rather than computing g_k^i for each triangle Δ_k^* at server side, one may consider estimating the global contribution of geometric errors g_k^i , which we denote as σ_G^2 at the server side and send it to the client. The value of σ_G^2 can be estimated by a priori considerations coming from the precision of G_{client} due to the precision of the acquisition and modelling instruments used to obtain it: for instance, if one uses a range camera with resolution characterized by a given σ , for example $\sigma = 100\mu$, one could simply approximate σ_G^2 with σ^2 .

Once σ_G^2 is received by the client it could be used to compute an approximate value g_k^i which depends on the requested view V^* and on V^i .

As shown in Fig. 11.15, as a simplification, if we assume that an error equally affects both x and y axes on images, the maximum geometric error happens when the lines of sight of V^* and V^i are orthogonal: that is the case when errors do not cancel each other. If they are parallel, the requested

Fig. 11.15 Geometric error \mathbf{e}_k on V^i and V^* .

view V^* is almost equal to V^i and the geometric error would not affect rendering significantly.

Note that the only characteristic of Δ_k affecting the norm of vector $(\mathbf{e}_{kx}^*, \mathbf{e}_{ky}^*)$, which enters Equation 11.28, is the distance of Δ_k from V^* , since in the projective transformation the size of the projection is inversely proportional to the distance from the object. Therefore, to avoid computation of g_k^i for each triangle, for surfaces of limited depth one could consider an average 3D model depth Δ' and approximate all the g_k^i as:

$$g_k^i = g^i = \frac{\langle \mathbf{n}^i, \mathbf{n}^* \rangle k \sigma_G^2}{d(\Delta', V^*)} \quad (11.29)$$

where $d(\Delta', V^*)$ is the distance between Δ' and V^* ; \mathbf{n}^i the line of sight of V^i and \mathbf{n}^* the line of sight of V^* ; and k is a constant value.

Approximation (11.29) is certainly inadequate when the surface depth is large or in general when some portions of the scene are very close and others very far from V^* . When this is not the case, parts of the 3D model can be grouped together, and a Δ' is associated to each group.

Note also that our surface model does not account for the illumination-dependent effects of shading and reflection, which are more relevant when



Fig. 11.16 Santa Claus: pictures $V^i, i = 0 \dots 3$.

the rendered view V^* is far from original views V^i . We can expect a second distortion contribution which grows with the deviation between the orientation of views V^* and V^i . Ignoring specularity, we expect this distortion term to be proportional to the signal power, suggesting the following augmented overall error:

$$D_{\mathbf{b},k}^* = \sum_{\mathbf{b}' \in B'} W_{\mathbf{b}' \rightarrow \mathbf{b},k}^{*i_{\mathbf{b},k}} \left(D_{\mathbf{b}',k}^{*i_{\mathbf{b},k}} + \left[g_k^{i_{\mathbf{b},k}} |\omega_{\mathbf{b}'}|^2 + g(\langle \mathbf{n}^{i_{\mathbf{b},k}}, \mathbf{n}^* \rangle) \right] E_{\mathbf{b}',k}^{i_{\mathbf{b},k}}(\Delta_k^{i_{\mathbf{b},k}}) \right) \quad (11.30)$$

where \mathbf{n}^i and \mathbf{n}^* are the surface normals to V^i and V^* respectively, and in the absence of careful modeling, $g(x)$ is set to $\gamma \tan(\cos^{-1} x)$, where γ determines the value we place on illumination fidelity.

Expression (11.30) explicitly formalizes the impact on V^* of the various sources of error: limited image quality at client side, geometric inaccuracies at client side and illumination, warping within DWT analysis and synthesis.

11.7 Experimental results

This section presents some experimental results about the performance of the RV-RD system.

Fig. 11.16 shows four original images $V^i, i = 0, \dots, 3$ of an object (Santa Claus), taken 90 degrees apart. Assume that the four images are sent to the client together with geometry information G , given by the 1000 triangles mesh as shown in Fig. 11.17 and that a new image V^* is requested. Fig. 11.18(a) shows the result of stitching without DWT as described in Section 11.4, while the other pictures of Fig. 11.18 show the results of stitching by DWT according to policy function f_{choice} based on distortion expressions (11.25) and (11.30) respectively.



Fig. 11.17 Santa Claus: geometric information G .

In order to interpret the results it may be useful to recall that in the case of distortion (11.25), if the images available at the client have all the same (good) quality, the stitching source is the one for which each triangle is more parallel to the viewer, i.e., the triangle with greatest ratio $|\Delta_k^i| / |\mathcal{T}_k^i(\Delta_k)|$. The chosen triangles in this case are shown in Fig. 11.18(b): different colors indicate different sources V^i . Fig. 11.18(c) shows the results of wavelet fusion with $D = 3$. If f_{choice} is based upon distortion expression (11.30) the stitching sources are shown in Fig. 11.18(d) and the wavelet fusion with $D = 3$ in Fig. 11.18(e). Some details (like the internal part of the left hand) cannot be reproduced since they are missing from the set of images V^i . Fig. 11.18(f) shows a photograph taken from the same position as V^* . Such a picture of course was not included in the set $V^i, i = 0, 1, 2, 3$.

The rendered views of Fig. 11.18 show that rendering according to a stitching policy based on (11.25), Fig. 11.18(c), gives better results than stitching without DWT, Fig. 11.18(a). As expected a policy based on (11.30) which considers also geometry errors, Fig. 11.18(e), performs better than (11.25) and gives results close to real photographs. It is also worth noting that the triangles selections corresponding to (11.25) and (11.30) are rather different.

Consider also the case in which only three of the images of Fig. 11.16 are available at the client with similar good quality, and one, shown in Fig. 11.19(a), at low quality. The results of stitching according to (11.30)

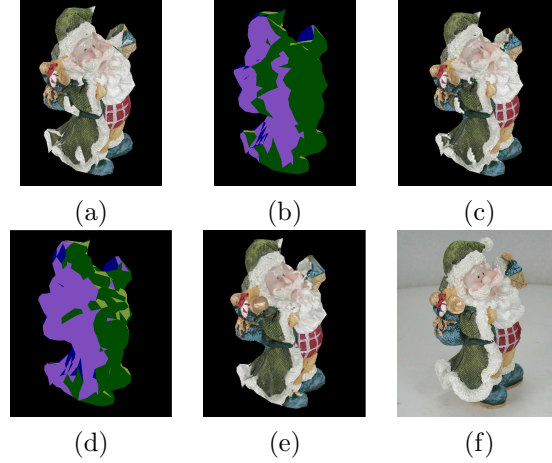


Fig. 11.18 Santa Claus from images with the same quality a) with stitching without DWT; b) stitching source and c) stitching based on DWT and distortion (11.25); d) stitching source and e) stitching based on DWT and distortion (11.30); f) a photograph taken from the rendering position.

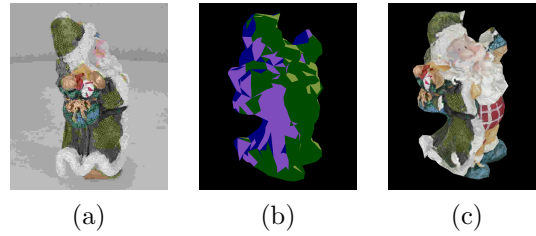


Fig. 11.19 Santa Claus from images with different quality a) the low quality image; stitching based on DWT and distortion (11.30); b) selected triangles; c) rendered view.

are shown in Fig. 11.19, namely the triangles selection according to (11.30) is shown in Fig. 11.19(b) and the result of the wavelet fusion with $D = 3$ is shown in Fig. 11.19(c). Such results should be compared with those of Fig. 11.18(d) and Fig. 11.18(e), respectively. The triangle selection of Fig. 11.19(b) shows an increase in the number of green and blue triangles with respect to Fig. 11.18(d). Purple triangles are chosen when no other source is available or when the image Fig. 11.19(a) is still the best source, in spite of its quality. The quality of the rendering of Fig. 11.19(c) is lower than that of Fig. 11.18(e), but is still of good quality.



Fig. 11.20 a) V^1 (good quality); b) V^2 (poor quality).

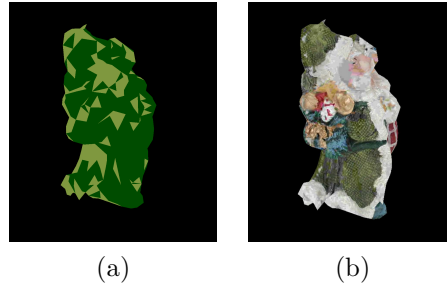


Fig. 11.21 Stitching from V^1 and V^2 : a) selected triangles; b) rendered view V^* .

The impact of image quality on the choice of the triangles is more evident if the points of view of V^i are close, since in this case the orientations of the triangles with respect to V^* are rather similar and there is a large number of triangles essentially offering the same information. Triangle selection in this case can exploit image quality. Let us assume that the client has available the two images V^1 and V^2 as shown in Fig. 11.20, of which V^2 has a much lower quality than V^1 .

Fig. 11.21(a) and Fig. 11.21(b) show the choice of triangles according to (11.30) and the rendered view V^* (between V^1 and V^2 , a little closer to V^2) with $D = 3$. Clearly the majority of the triangles is selected from V^1 (the dark green ones at the right), since it is the best quality view. Some parts were selected from the low quality view: probably the right cheek of Santa Claus comes from V^1 because it is a rather homogeneous region with a small distortion value with respect to the original view.

Finally assume that the client, after some delay, besides V^1 has available a new view $V^{2'}$ shown in Fig. 11.22(a), of quality higher than that of V^2 and

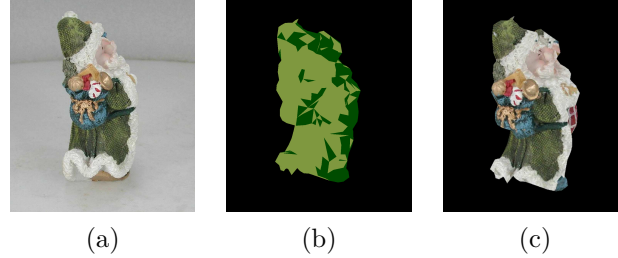


Fig. 11.22 Stitching from V^1 and $V^{2'}$: a) $V^{2'}$; b) selected triangles; c) rendered view V^* .

comparable with that of V^1 . In this case (11.30) leads to an f_{choice} giving the triangle selection and the rendered view V^* shown in Fig. 11.22(b) and Fig. 11.22(c) respectively. As expected, distortion (11.30) in this case takes a large number of triangles from $V^{2'}$ and less triangles from V^1 than in the previous case.

11.8 Summary

This chapter introduced a new framework for the remote visualization of 3D models, for convenience called RV-RD, where the geometry and texture information available at server side is incrementally delivered to the client according to a server policy and the client has its own policy for using the available information for best rendering results. It is important to note that the server and client policies are assumed to be independently optimal, i.e., the server does not need to know what the client does in order to decide what to send; and the client, independently of what the server does, uses at best the information it has available.

The contribution of this chapter besides the overall RV-RD scenario is the presentation of a client policy based on a distortion measure accounting for the possible limited quality of the images and the possible errors in the geometry available at the client. The proposed client policy rests on strong theoretical considerations and the experimental performance confirms the expectations.

It may be worth observing that the client policy is the reasonable starting point for the implementation of an RV-RD system, since this is what directly controls the visualization quality. The implementation of a RV-RD system requires an effective server policy which can be supported by any

reasonable scheme for transmitting the information available at the server.

Our current research explores server policy with the fundamental issues related to how the server should distribute available transmission resources amongst the various original view images and geometry informations needed by the client to render a new view. Included in this question is whether the server should transmit elements from an entirely new original view image, which is more closely aligned with the requested view, rather than refining nearby original view images of which the client already has a low resolution version. It is appropriate to relate such issues to more general considerations concerning sampling and quantization of the plenoptic function to which they are deeply connected.

As pointed out in the introduction, the RV-RD system approach offers a new intriguing conceptual environment for addressing basic fundamental questions about the visual impact of geometry and texture as well as practical solutions enabling the remote visualization of 3D models.

Bibliography

- Cheng, I. and Basu, A. (2004). Reliability and judging fatigue reduction in 3d perceptual quality, in *The 3rd International Symposium on 3DPVT* (IEEE).
- D.Taubman and R.Prandolini (2003). Architecture, philosophy and performance of jpip: internet protocol standard for JPEG 2000, in *Int. Symp. Visual Comm. and Image Proc.*, Vol. 5150 (IEEE), pp. 649–663.
- Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F. (1995). *Computer Graphics Principles and Practice* (Addison-Wesley).
- Garland, M. and Heckbert, P. (1997). Surface simplification using quadric error metrics, URL sherry.ifi.unizh.ch/garland97surface.html.
- Guskov, I., Sweldens, W. and Schröder, P. (1999). Multiresolution signal processing for meshes, *Computer Graphics Proceedings (SIGGRAPH 99)*, pp. 325–334.
- Hartley, R. and Zisserman, A. (2000). *Multiple View Geometry in Computer Vision* (Cambridge University Press, Cambridge, United Kingdom).
- Hoppe, H. (1996). Progressive meshes, *Computer Graphics* **30**, Annual Conference Series, pp. 99–108, URL citeseer.csail.mit.edu/hoppe96progressive.html.
- Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P. and Scopigno, R. (2004). Protected interactive 3d graphics via remote rendering, in *proc. SIGGRAPH 2004* (IEEE).
- Levoy, M. (1995). Polygon-Assisted JPEG and MPEG compression of synthetic images, in *Proceedings of SIG-GRAPH Computer Graphics Proceedings, Annual Conference Series*, pp. 21–28.
- Rusinkiewicz, S. and Levoy, M. (2000). QSplat: A multiresolution point rendering system for large meshes, in K. Akeley (ed.), *Siggraph 2000, Computer Graphics Proceedings* (ACM Press / ACM SIGGRAPH / Addison Wesley Longman), pp. 343–352, URL citeseer.csail.mit.edu/rusinkiewicz00qsplat.html.
- Schroeder, W. J., Zarge, J. A. and Lorensen, W. E. (1992). Decimation of triangle meshes, *Computer Graphics (SIGGRAPH '92 Proc.)* **26**, 2, pp. 65–70.
- Taubman, D. and Secker, A. (2003). Highly scalable video compression with scalable motion coding, *Proceedings of International Conference on Image Pro-*

cessing **3**, 3, pp. 273–276 v.2.

Taubman, D. S. and Marcellin, M. W. (2002). *JPEG2000 Image compression fundamentals, standards and practice* (Kluwer Academic Publisher, Cambridge, United Kingdom).

Zanuttigh, P., Brusco, N., Taubman, D. and Cortelazzo, G. (2005). Greedy non-linear optimization of the plenoptic function for interactive transmission of 3d scenes, *International Conference of Image Processing ICIP2005, Genova*.