TEXTURE MAPPING FOR 3D BUILDING MODELS

by

Hakan Karlıdağ
B.S, in Computer Engineering, Boğaziçi University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Systems and Control Engineering
Boğaziçi University
2010

TEXTURE MAPPING FOR 3D BUILDING MODELS

APPROVED BY:

Prof. Lale Akarun (Thesis Supervisor)	....
Dr. Ali Vahit Şahiner	....
Assoc. Prof. Cem Yalçın	....

DATE OF APPROVAL: 22.04.2010

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, Prof. Lale Akarun for her guidance, support and comments during the preparation of this thesis.

I also would like to extend my gratitude to Dr. Ali Vahit Şahiner and Assist. Prof. Cem Yalçın for participating into my thesis jury and for their valuable comments.

I also would like to express my special gratitude to Koray Balcı who gave support and motivation since the very beginning of this research.

I would like to thank to all my teachers in CMPE who have influenced me throughout my education.

Finally, with greatest thanks to Gülben Yılmaz with whom I feel very lucky to have met in my life.

ABSTRACT

TEXTURE MAPPING FOR 3D BUILDING MODELS

The need and attempt for creating mathematically-defined three dimensional (3D) models of real world objects has a long history. While, in the past, creating the model was a problem itself, with recent developments in 3D reconstruction, creating accurate, photorealistic and photogrammetric 3D models of the objects has become the focus point. Starting from the architectural models, an interactive texture mapping system was developed concentrating on the visual appearance of the predefined models. The system presents a semi-automatic way of extracting, correcting, and mapping the appropriate textures to the given 3D building models using the images obtained by standard consumer-level digital cameras. Projective geometry takes place in extraction and rectification of texture candidates while popular graph cut optimization approach was utilized to create seamless texture composites using these candidates. A refinement phase was adopted for this procedure with a series of refinement tools including Poisson image editing in terms of seamless cloning. For testing purposes some publicly available datasets were used besides the imagesets that were created by photographing real world objects. It was shown that most buildings could be textured in an acceptable photorealistic quality without any predefined information about the datasets. Furthermore, it was observed that the texture mapping times, even for detailed building models, were quite low. This work mainly focuses on close-range or ground level imagery since the aim is to create detailed and high quality photorealistic view of the models. However, the approach could easily be extended towards the needs of aerial imagery or large scale reconstruction.

ÖZET

3B BİNA MODELLERİ İÇİN DOKU EŞLEME

Gerçek objelerin üç boyutlu (3B) matematiksel modellerini oluşturma ihtiyacının ve girişimlerinin uzun bir geçmişi var. Eskiden modelin kendisini oluşturmak başlı başına problemin kendisi iken, 3B geriçatma teknolojisindeki gelişmelerle, hassas, fotogerçekçi ve fotogrametrik modeller oluşturmak daha önemli hale geldi. Bu çalışmada önceden tanımlı mimari modellerin görünümüne odaklanarak interaktif bir doku eşleme sistemi geliştirdik. Sistem, standart tükeci fotoğraf makineleri ile çekilen fotoğraflar kullanılarak verilen 3B model için uygun dokuların ayıklanması, düzeltilmesi ve eşlenmesi için yarı-otomatik bir yöntem önermektedir. Doku adaylarının ayıklanması ve düzeltilmesinde izdüşümsel geometri kullanılırken, bu adaylar kullanılarak kesintisiz doku bileşimleri oluşturmak için popüler grafik kesme eniyileme yöntemi kullanılmıştır. Bu süreci kesintisiz klonlamada kullanılan Poisson imge işleme yönteminin de aralarında olduğu bir çok yardımcı aracın bulunduğu bir iyileştirme aşaması takip etmektedir. Test amacıyla kendi oluşturduğumuz imge setlerinin yanı sıra bazı kullanıma açık veri setlerini de kullandık. Yapılan denemeler sonucu veri setleriyle ilgili herhangi bir önbilgi kullanmadan binaların hemen hepsine makul fotogerçekçi kalitelerde doku eşleşmesi yapıldığını gözlemledik. Ayrıca, doku eşleştirme zamanlarının, detaylı bina modelleri için bile, oldukça düşük olduğunu gösterdik. Bu çalışmada modellerin detaylı ve yüksek kalitede fotogerçekçi görünümlerini elde etmeyi amaçladığımız için yakın ölçekli ve zemin seviyesinden fotoğraflama yöntemini temel aldık. Fakat, kullanılan yöntem kolayca havadan ve geniş ölçekli geriçatma sistemlerini de içine alacak şekilde genişletilebilir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

a	Source node on a graph
$a_{\{p,q\}}$	An auxiliary node between nodes p and q
A	Subset of graph
b	Sink node on a graph
B	Subset of graph
C	A cut on a graph
d	Distance
D	Graph cut data penalty term
D_p	Graph cut data penalty at pixel p
$e_{\{p,q\}}$	An edge between nodes p and q
$E(f)$	Graph cut energy functional with particular label image f
E_{data}	Graph cut data penalty term
E_{smooth}	Graph cut interaction/smoothness penalty term
f	Label image
$\hat{f}$	Lowest energy labeling within a single α-*expansion* move
f_p	Image label at pixel p
g	Function representing source image in seamless cloning
G	Graph
H	Projective transformation matrix
H_t	Scalar edge potential between two neighboring pixels of image t
I_{inf}	The line at infinity
I_k	Image with label k
I_p	Observed intensity at pixel p
L	Finite set of labels
N	Set of interacting pairs of pixels
p	Pixel position
P^2	Projective space
$\mathbf{P}$	Partition

S	Image definition domain
$t_p^{\bar{\alpha}}$	A t-link between terminal α and node p
v	Set of vertices on a graph
$\mathbf{v}$	Guidance vector field
V	Graph cut interaction/smoothness penalty term
$V_{p,q}$	Graph cut interaction/smoothness penalty between pixels p and q
w	Weight term
α	Label of interest in graph cut optimization
θ	Angle between the plane normal and viewing camera angle
ε	Set of edges on a graph
μ	Median color value
Ω	A closed subset of image definition domain
$\partial\Omega$	Boundary of Ω
∇	Gradient operator
Δ	Laplacian operator

2D	Two Dimensional
3D	Three Dimensional
HDR	High Dynamic Range
ICM	Iterated Conditional Modes
LNF	Luminance-Normalized Facade
MRF	Markov Random Field
PDE	Partial Differential Equation
RGB	Red Green Blue
VP	Vanishing Point
XML	Extensible Markup Language

1. INTRODUCTION

Accurate 3D models of buildings are needed for a variety of applications, such as fly-through or walk-through rendering, simulation for mission planning, modeling for augmented reality environments, industrial rapid prototyping systems, equality control purposes, faithful reconstruction of cultural heritage, computer gaming and town planning. Therefore, the problem of 3D modeling from images and video has received a large interest in the computer graphics and vision communities. A large body of research exists in this area however detailed analysis of facade texture and microstructure has been very limited.

1.1. Motivation

Recent researches and developments in 3D reconstruction of architectural models have given rise to the need of photorealistic models. With the growing demand in this area the quality of model views become significant day by day. Hence, in most of the 3D reconstruction applications, it is desirable to capture not only the geometry, but also the visual appearance of an urban environment. Shading models with pseudo textures is a commonly used approach, but does not represent true building textures. For realistic texture mapping of 3D building models, using images acquired with digital cameras is more beneficial because they provide high visual realism as well as cultural and functional information about the building.

In an urban area, while capturing surface appearance of real world objects, it is hard to find ground views that capture an entire building. Since many buildings are close to each other, and narrow streets limit the field of view, extraction of textures in an acceptable way becomes a quite difficult problem. Thus, researchers overcome this problem by using two or more photos of a building which brings a number of refinement techniques to the area of texture mapping. Moreover, since the building appears in multiple images, automatic determination of optimal texture with no distortion and least occlusion for each wall of the building becomes very significant for high quality and effective photorealistic 3D visualization.

For generating photorealistic, textured, planar 3D models of architectural structures from an unordered collection of photographs, we develop an interactive system. The system presents a semi-automatic way of extracting, correcting, and mapping the appropriate textures to the given 3D building models using the images taken by standard consumer-level digital cameras. This work mainly focuses on close-range or ground level imagery since the aim is to capture detailed and high quality photorealistic view of the models. However, the approach can easily be extended according to the needs of aerial imagery or large scale reconstruction.

In order to achieve texture mapping in our system, firstly the user draws outlines overlaid on 2D photographs which specifies the planes to be extracted as textures. The user can also display the extracted planes instantly in order to decide whether or not to use as candidate input photographs. Then, seamless texture maps are automatically generated by combining multiple input photographs using graph cut optimization and Poisson blending. These two approaches are the main tools that were used in order to create texture composites and apply a refinement over this composites, respectively.

Graph cut optimization is the preferred technique used in computation photography for finding optimal seams between image regions being stitched together while Poisson blending is used to reduce or remove any visible artifacts that might remain after the image seams are joined.

1.2. Literature Review

1.2.1. 3D Reconstruction

The problem of 3D modeling from images has received a big interest in the photogrammetry, computer graphics and vision communities. Texture mapping contributes to the 3D modeling task as a visualization phase and mostly 3D modeling approach itself defines the methods that would be applied during texture mapping process. In fact, the level of detail in visualisation and the amount of user interaction that is needed are highly dependent on the approach used for 3D modeling. For instance, ground based and close-range methods uses improved rectification and texture modification techniques even with texture synthesis algorithms; whereas in reconstruction techniques, which uses aerial imagery, texture mapping is generally composed of basic image rectification and warp operations caused by smoothing algorithms applied to 3D model. Therefore, while exploring texture mapping approaches it highly important to understand recent concepts developed in 3D reconstruction.

3D reconstruction algorithms are classified in different manners according to different point of views. Here, the most basic classifications and significant works that were done according to these classifications are described.

1.2.1.1. Geometry-Based vs. Image-Based. According to an early research [1] there were two types of modeling and rendering architecture. The geometry-based approach places the majority of the modeling task on the user, whereas the image-based approach places the majority of the task on the computer. Yet, hybrid approaches divide the modeling task into two stages, one that is interactive and one that is automated. Some studies also named this seperation as *Automated vs. Interactive.* Significant success has been recently reported with fully automated systems such as [2], [3]. These systems are based on structure from motion process to first recover the camera poses and a sparse point cloud reconstruction of the scene. From the sparse reconstruction, dense multi-view stereo algorithms could generate a dense mesh model. While systems such as [2] process video, [4] use improved feature extraction and matching techniques to make structure from motion work with unordered photo collections obtained from the internet. In addition to these methods, [5] proposes a fully automatic method that uses generative models for buildings, but it is restricted by the prior models and can only operate on small sets of images.

Although these works are impressive, yet they require dense photo collections and their quality may suffer if the camera motion is degenerate or the scenes lack sufficient textures. These limitations are overcome by having a user in the loop to interactively guide the geometry creation. Most recent examples of such interactive systems are described in [1,7–12]. Facade [1] is one of the earliest hybrid modeling systems designed for modeling architectural scenes which later gave rise to a commercial product called Canoma. It provides a set of parameterized 3D primitives to user in order to model a part of the scene. Whereas, [10], instead of using a set of pre-defined shapes, proposed a user guided method for creating and re-using building blocks for adding in geometric detail once a coarse model has been generated. On the other hand, single-view modeling techniques such as [9, 13] and other methods such as [7, 14] have used vanishing point constraints in modeling architecture. Finally, [11] and [12] are the most recent and quite impressive systems which significantly decrease the reconstruction times and provides functional user interfaces.

1.2.1.2. Active Camera vs. Passive Camera. A natural choice to satisfy the requirement of modeling the geometry and appearance is the combined use of active range scanners and digital cameras. Frueh and Zakhor [15], Akbarzadeh et al. [16], and Pollefeys et al. [2] used such a combination, capturing large amounts of data in continuous mode, in contrast to the previous approaches [17] that captured a few isolated images of the scene from a set of pre-specified viewpoints. They used laser scanners which have the advantage of providing accurate 3D measurements directly. On the other hand, they can be cumbersome and expensive. Several researchers in photogrammetry and computer vision address the problem of reconstruction relying on passive sensors (cameras) in order to increase the flexibility of the system while decreasing its size, weight and cost.

1.2.1.3. Large Scale vs. Small Scale. There have been significant efforts in large scale reconstruction, typically targeting urban environments or archeological sites. These researches on urban reconstruction could be both from ground-based imagery [2, 16] and from aerial images [15]. However, small scale reconstruction generally needs more detailed work and uses ground-based imagery [12, 18, 19].

Large scale systems typically generate partial reconstructions which are then merged. Conflicts and errors in the partial reconstructions are identified and resolved during the merging process.

1.2.2. Texture Mapping

1.2.2.1. Texture Mapping in General. Methods for building texture retrieval mentioned in the literature are mostly grouped in three general classes: (1) texturing building tops using top-down views of the roofs, typically, airborne aerial images [15,20], (2) texturing facades by utilizing ground-based images [12,18,19,21], and (3) their combinations [22]. While top-down aerial image is capable of efficiently providing a complete set of the roof shapes of all buildings at sufficient detail and accuracy, the vertical walls of buildings are usually invisible in normal horizontal aerial image. On the other hand, ground-based data acquisition systems are capable of providing building facades as seen from the street level, but the time required to extract these facades from the images increase with level of detail and quality. Moreover, the roofs of building are not accessible from the ground-based acquisition system if the building is too high.

Another popular approach, especially for displaying open and wide spaces, was using panoramic views created from stitched photographs [23–25]. "Route panorama" [26] technique followed these works to provide photo-realistic street views. However, although panoramic visualization could generate impressive scenes, it displays only facial appearances instead of modelling accurate building textures. Moreover, original texture images for building facades may have shadows or portions blocked by foreign objects and panoramic views lacks in handling this kind of problems.

In some systems the camera parameters are known and geometric distortions caused by viewing angle can be corrected using photogrammetric methods of perspective photo mapping [27]. However, most commodity digital cameras do not provide complete viewing parameters unless augmented with additional (often expensive) equipment such as GPS, INS, and digital compass. Moreover, although we have the camera parameters we need expert users in order to integrate camera information into texture mapping systems. Therefore, since in an interactive system these kind of approaches can be cumbersome for ordinary users to take place in the loop, this kind of systems generally includes automated techniques.

For images with unknown viewing parameters, the geometries of buildings in images might be approximated if geometric constrains are well known, but photogrammetric corrections are difficult. In order to estimate related camera parameters, [23] used correlations of overlapped images. Vision-based modelling methods were also suggested for obtaining relative pose between the cameras and 3D scene geometry from motion imagery [28]. One of the recent and fairly robust methods is to reconstruct by computing vanishing points (VP) of building line segments in images. While [29] performs stable VP estimation in a single image, [4,12] extended the concept by jointly estimating VPs in multiple images of the same scene. On the other hand, [1] identified building boundaries from images to determine facets of the building and to map corresponding texture blocks to model surfaces from cropped areas selected from an image spool.

That is to say, texture mapping problems highly differs according to the goal and scope of the texture mapping applications. In the case of aerial imagery, which is mostly used in cyber city applications, optimal vertical textures are simply obtained by using ortho-rectification after projective mapping [15,20]. However, when reconstruction of specific buildings are considered we generally need more detailed and accurate systems with additional processes such as texture selection, occlusion removal and texture refinement [12,18,19]. In other words, large scale systems tend to be more automated systems with less amount of user interaction which relatively decreases the level of detail, while small scale systems generally include more interaction to increase the level of detail and quality.

1.2.2.2. Creating Texture Composite. In an urban area, it is hard to find ground views that capture an entire building since many buildings are close to each other, and narrow streets limit the field of view. The small field of view prevents the extraction of textures in an acceptable way. So researchers overcome this problem by using two or more photos of a building which brings a number of refinement techniques to the area of texture mapping. Moreover, because the building is appeared in multiple images, automatic determination of optimal texture with no distortion, least occlusion and high resolution for each wall of building becomes very significant for high quality, efficiency and effectiveness of photo-realistic 3D visualization of building model. Two main approaches are used at this point. Some researchers, generally the ones using large numbers of images, select necessary textures according to the rendering view point [1], [30], [31], [32], [33]. This approach utilizes images captured from a collection of viewpoints and uses them to generate specific textures for different rendering views. The second approach, on the other hand, applies a preprocessing phase in order to select the best textures or texture composites that will be used during the entire rendering process [34], [35], [36], [18], [12]. While specifying the optimal textures [18] simply weighted the texture fragments according the angle between the viewing direction during acquisition and the surface normal of the related facade. Fragments are then blended together according to their weights and formed the necessary textures. Further works like [12] extended these list of decision parameters by adding parameters like color consistency, visibility etc. and used an improved optimization technique called graph cut optimization to create texture composites according to this parameters.

Since individual digital texture photographs are usually taken at different viewing conditions (view points, looking angles, zoom factors etc.), they are of assorted perspectives, scales, brightness, contrasts, colour shadings and other properties that are significant in imagery. These variations need to be adjusted in order to integrate into a seamless mosaic. Therefore, the first challenge of 3D building texture mapping is to merge images pertaining to the same building facade into a complete texture mosaic that is continuous in geometric outlines and in colour shadings. Image mosaicking is a common technique used in a variety of related applications to generate complete texture information of a building facade from digital photographs. One of the techniques for generating image mosaics of the real-world environment was to merge sequences of video frames [37], [18]. However, general image mosaicking algorithms were not designed for photo-realistic texture mapping and generally did not fulfil the requirements of creating complete and seamless building texture information, which is critical in city and building visualization and applications. In addition, directly using video sequences for building texture often requires intensive manual treatments to eliminate occlusions, blurs and other artifacts [38].

Another approach for creating texture composites was to define the problem as an energy minimization problem by using some quantities (such as intensity or disparity) changing over the images. Energy-based methods attempt to model some global image properties that cannot be captured, for example, by local correlation techniques. The main problem, however, is that interesting energies are often difficult to minimize. Due to this inefficiency of computing the global minimum, many authors have opted for a local minimum. An example of a local method using standard moves is Iterated Conditional Modes (ICM), which is a greedy technique introduced in [39]. For each pixel, the label which gives the largest decrease of the energy function is chosen, until convergence to a local minimum.

Another example of an algorithm using standard moves is simulated annealing, which was popularized by [19]. Unfortunately, it requires exponential time and as a consequence it is very slow. Theoretically, simulated annealing should eventually find the global minimum if it runs for long enough but [20] demonstrate that practical implementations of simulated annealing may give results that are very far from the global optimum. Sampling algorithms that were developed [40] could make larger moves in order to improve the rate of convergence of simulated annealing.

If the energy minimization problem is discussed in continuous terms, variational methods that use Euler equations, which are guaranteed to hold at a local minimum, can be applied [41]. On the other hand, another alternative is to use discrete relaxation labeling methods like [42], [43]. In relaxation labeling, combinatorial optimization is converted into continuous optimization with linear constraints. Then, some form of gradient descent is used.

In some cases, global minimum can also be computed via dynamic programming

[44]. However, as stated in [45], in general, the two-dimensional energy functions that arise in early vision cannot be solved efficiently via dynamic programming.

Recently, the approach of representing the quality of pixel combinations as a Markov Random Field and formulating the problem as a minimum cost graph cut became popular. The former examples of this approach was stated by [46], [47], [48]. These works used graph cuts to find the exact global minimum of certain type of energy functions. However, these methods apply only if the labels were one-dimensional and their energies were not discontinuity preserving. More recently, [45] contribute to the area by two new algorithms for multidimensional energy minimization that use graph cuts iteratively. They achieve approximate solutions to this NP-hard minimization problem with guaranteed optimality bounds and most importantly they generalize the previous results by allowing arbitrary label sets, arbitrary data terms and a very wide class of pairwise interactions that includes discontinuity preserving cases.

After this generalization, graph cut optimization method, which is also detailed in this thesis in Section 2.2, has been used for a variety of tasks, including image segmentation, stereo matching and optical flow. For instance, [49] introduced the use of graph-cuts for combining images. Although they mostly focused on stochastic textures, they showed the ability to combine two natural images into one composite by constraining certain pixels. Then, [50] extend this approach to the fusion of multiple source images using a set of high-level image objectives.

Although graph cut optimization mostly finds good seams between image composites, in some cases results will not be satisfactory and researchers apply an additional refinement operation on the results. One of the techniques employed for such a purpose was Poisson image editing which is also the main tool applied for refinement in this thesis. Related works stated on this issue is given in the next section (Section 1.2.2.3) and mathematical details will be discussed in Section 2.3. As far as we know, [50] is the first one to use Gradient Domain Fusion in combination with graph cut optimization for such a purpose. They build up a new interactive photomontage method with new cost functions that extend the applicability of graph cuts to a number of new applications. [12] followed it and transferred the concept to the space of 3D reconstruction and texture mapping.

1.2.2.3. Texture Refinement. We implement a specific type of gradient domain fusion approach which makes use of Poisson image editing techniques. Poisson equation has been used extensively in computer vision. In the specific context of image editing applications there are a number of works related to the use of the Poisson equation proposed here.

The earliest and well-known work in image fusion used Laplacian pyramids and per-pixel heuristics of salience to fuse two images [51, 52]. These early results demonstrated the possibilities of obtaining increased dynamic range and depth of field, as well as fused images of objects under varying illumination. However, these earlier approaches had difficulty capturing fine detail. They also did not provide any interactive control over the results.

In [53], the gradient field of a High Dynamic Range (HDR) image is rescaled non-linearly, producing a vector field that is no longer a gradient field. A new image is then obtained by solving a Poisson equation with the divergence of this vector field as right-hand-side and under Neumann boundary conditions specifying that the value of the gradient of the new image in the direction normal to the boundary is zero. In contrast, the method proposed in [54] can be applied to arbitrary patches selected from an image, not just to the entire image. In order to do this, Neumann boundary conditions on a rectangular outline is replaced by Dirichlet conditions on an arbitrary outline.

Elder and Goldberg [55] introduced a system to edit an image via a sparse set of its edge elements (edgels). To suppress an object, associated edgels are removed; to add an object, associated edgels as well as color values on both sides of each of these edgels are incorporated. The new image is then obtained by interpolating the colors associated to the new set of edgels and this stands for solving a Laplace equation (a

Poisson equation with a null right hand side) with Dirichlet boundary conditions given by colors around edgels. However, generally, editing edgels and associated colors is not simple. In addition, image details are lost when converting to and from the contour domain, which might be undesirable.

Lewis [56] described a high quality method for separating intensity-detail from overall image region intensity. Here, spots are removed from fur images by separating out the brightness component from details in a selected region and replacing the brightness by harmonic interpolation (solving a Laplace equation) of the brightness at the selection boundary.

In terms of image editing functionalities, two existing techniques achieve seamless cloning as the the system introduced by [54] which is also implemented in our work. The first one is Adobe Photoshop 7's Healing Brush [57]. As far as we know, the technique used by this tool has not been published yet, so we don't know whether it uses a Poisson solver or not. Mostly this kind of standard image-editing tools require manual selection of boundaries, which is time consuming and burdensome. In contrast, this thesis offers a smart blending tool which makes use of texture composite results to achieve automatic identification of editing regions and boundaries.

The second technique is the multiresolution image blending proposed in [58]. The idea is to use a multiresolution representation, namely a Laplacian pyramid, of the images of interest. The content of the source image region is mixed, within each resolution band independently, with its new surrounding in the destination image. The final composite image is then recovered by adding up the different levels of the new composite Laplacian pyramid. The technique results in multiresolution mixing where finest details are averaged very locally around the boundary of the selection. This fast technique achieves an approximate insertion of the source Laplacian in the destination region whereas this Laplacian insertion is performed via the solution of a Poisson equation in this work. More importantly, multiresolution blending takes data from distant source and destination pixels, through the upper levels of the pyramid. This long range mixing, which might be undesirable, does not occur in the technique we applied.

[59] further presented a pyramid blending algorithm that used different alpha masks in different bands. [60] developed two complex image stitching algorithms in the gradient domain to optimize the mosaicking quality. These are all effective methods, but they all require intensive computation and most of them can only deal with a pair of input images at a time.

After Perez et. al. [54], Agarwala et. al. [50] used the same approach in which a region of a single source image is copied into a destination image in the gradient domain. However, their work differs in that they copy the gradients from many regions simultaneously, and they have no single destination image to provide boundary conditions. Thus, the Poisson equation must be solved over the entire composite space. This approach is useful but needless in our case since a one to one correspondence between the source and destination is sufficient to reach the desired results.

Finally, since the proposed guided interpolation framework was implemented in [54], in the case of seamless cloning, various interpolation methods have been proposed to fill in image regions automatically using only the knowledge of the boundary conditions. One class of such approaches is composed of inpainting techniques [61] where partial differential equation (PDE) based interpolation methods are introduced. The PDEs to be solved are more complex than the Poisson equation, and work only for merging fairly narrow gaps. Example-based interpolation methods [62, 63] where the new image region is synthesized using an arrangement of many small patches are an alternative to inpainting. These methods handle large holes and textured boundaries in a more successful way.

Apart from gradient domain methods, other refinement techniques also exist. An alternative technique uses histogram matching or equalization to force colour and shading distributions of candidate images to be within the same range [64]. This technique is applied to minimize the color and shading differences between composites, however, directly applying this method to close-range images for texture generation may cause serious misrendering of colour shadings such as hazy or low-contrast images.

Another refinement method targets the degradation effects caused by the luminance. [9], present a filter that factors the image into a texture component and an illumination component which is also useful for relighting since the decoupled texture channel has a uniform level of illumination. Whereas, [36], normalize the facade images by linear gray-level stretching. The resulting luminance-normalized facade images (LNF images) have the same average luminance and thus are comparable to one another for the other phases.

1.2.2.4. Interface Tools. In some specific cases, texture composition results may be unsatisfactory even after automatic refinement tools. Thus, systems which employ user knowledge while texture mapping, additionally, include some handy tools in order to improve the quality of results and decrease the mapping time.

In [65], the user can draw strokes to indicate which object or part of the texture is undesirable. The corresponding region is automatically extracted and image inpainting [66] is used for automatically inpainting. Similarly, [12] offers an interactive way in which the user can specify additional constraints using brush strokes. Constraints could be both positive (the user indicates preference for specific portions) or negative (the user erases undesired portions). [9], on the other hand, used an increased level of interaction by implementing a Clone brushing (rubberstamping) tool for the seamless alteration of pictures. It interactively copies a region of the image using a brush interface and is often used to remove undesirable portions of an image, such as blemishes or distracting objects in the background. Moreover, 3D reconstruction systems that implement modeling interfaces also introduce some useful 3D tools. For instance, [12] provided standard CAD utilities like extrusion, plane completion, mirroring, while [11] implemented a mirror plane in order to build a complete model of an object which is symmetric about a plane.

1.3. Objective and Scope of the Work

Sinha et. al. [12] and Hengel et. al. [11] are recent successful examples of semiautomatic 3D reconstruction systems with built-in texture mapping modules. While Hengel et. al. focuses on video and structure from motion, Sinha et. al. decrease the level of user interaction by introducing many assistive tools in terms of texture mapping. However, in both systems, texture mapping comes after 3D modeling and make use of estimated camera parameters for each image.

In this thesis, a 2D system was proposed to produce photorealistic texture maps for 3D building models. Given digital photographs of a building and the corresponding 3D model, accurate facade textures are provided in a semiautomatic way. In this process, it is assumed that the whole process is carried out by ordinary users with no specific professional training in the area and ground view photographs, taken by standard cameras, are used. All perspective distortions existing in the photographs are removed automatically by texture rectification using projective geometry. Texture composites are then created using graph cut optimization according to these rectified texture candidates. In addition, a refinement phase follows this process so that users are able to perform some minor modifications over the texture composites. One of the most important refinement operations is seamless cloning which lets users to remove seams between texture partitions by using guided interpolation. Finally, all algorithms and assistive tools are presented in a user friendly interface to provide a complete visual experience for the users.

Using our system, after the user chooses the planar face to texture, he/she defines the planar area in the photographs by the help of a line tool. Specified areas are cropped and rectified automatically to make them proper texture candidates. After this point, the user can either use the texture composite created by using these candidates or perform some additional refinements over the recommended composite. Texture composites was created using graph cut optimization by applying the idea of [50] in digital montage applications.

Targeting the ordinary users, a user friendly application was developed in order to provide a complete visual experience with a series of utilized tools created to simplify the texture mapping process. For instance, seamless cloning approach [54] helps users quickly remove seams existing in the composites, while traditional flip, exclude and substitute tools let users gain time especially for the models including repeating or symmetrical structures. Additionally, a Harris corner detector [67] is implemented to specify plane borders more accurately. Finally, a 3D model viewer was created to let users visualize their mapping at any time of the process.

1.4. Outline of the Thesis

The organisation of the rest of the thesis is as follows: Chapter 2 starts with the methods used while extracting the candidate planar images from digital photographs. Here, necessary formulations are stated in order to remove projective distortion from the images. Then, in the second section of this chapter, graph cut optimization process is detailed with the necessary parameters and algorithms. Besides, desired optimization criteria are included in this section. Chapter 2 ends up with the refinement techniques implemented to improve final texture quality. Gradient domain fusion is the main tool explained in this section. Chapter 3, on the other hand, gives details about the user interface application which includes implementations of all features and methods stated in this thesis. Texture mapping examples with different kinds of buildings and the results of application usage are stated in Chapter 4. All results are interpreted in Chapter 5 and the thesis concludes with the possible extensions and contributions.

2. TEXTURE MAPPING

2.1. Image Rectification

Image rectification, concisely, is a transformation process used to project multiple images onto a common image surface. It is used in two main areas: (1) in computer stereo vision to simplify the problem of finding matching points between images; and (2) in geographic information systems to merge images taken from multiple perspectives into a common map coordinate system. Our system is more related to the latter one since we are not dealing with the reconstruction phase directly. We have the video sequences or images of a 3D object and we do not have any information about the camera parameters. So that, in order to begin the texture mapping process of this object we have to extract fronto-parallel views (i.e. parallel to the image plane) of each plane forming the 3D model. Figure 2.1 clearly exemplifies this method.

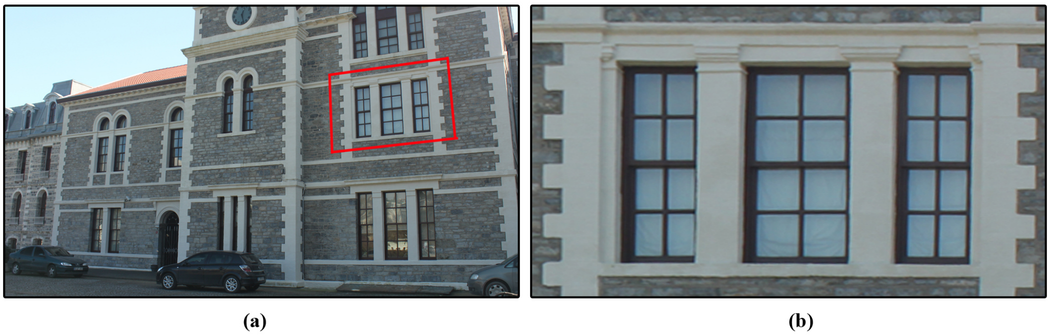


Figure 2.1. Image rectification. a) Original image. b) Rectified area.

Mathematically, while using the photos or video sequences of a building or object we will be dealing with perspective imaging and it is well known that under perspective imaging a plane is distorted since it is mapped to the image by a plane projective transformation. For instance, in Figure 2.1 the windows are not rectangular in the first image, although the originals are. To determine the transformation we are using projective geometry. Since the image is a projective distortion of the original, it is possible to "undo" this distortion by canceling projective transformation by computing the inverse transformation and applying it to the image. The result will be a new synthesized image in which the objects in the plane are shown with their correct geometric shape (Figure 2.1 (b)).

2.1.1. Removing the Projective Distortion

As stated in [68] once the transformation is determined, Euclidean measurements, such as lengths and angles, can be made on the world plane directly from image measurements. Furthermore, the image can be rectified by a projective warping to one that would have been obtained from a fronto-parallel view of the plane. So to determine the desired transformation we deal with the following notions.

2.1.1.1. Ideal Points and The Line at Infinity. We know that homogeneous vectors $x = (x_1, x_2, x_3)^T$ such that x_3 is nonzero correspond to finite points in R^2. Moreover, one may augment R^2 by adding points with last coordinate $x_3 = 0$. The resulting space is the set of all homogenous 3-vectors, namely the *projective space* P^2. The points with last coordinate $x_3 = 0$ are known as ideal points or points at infinity. The set of all ideal points may be written as $(x_1, x_2, 0)^T$, with a particular point specified by the ratio $x_1 : x_2$. Note that this set lies on a single line, the line at infinity, denoted by the vector $I_\infty = (0, 0, 1)^T$. Indeed one verifies that $(0, 0, 1)(x_1, x_2, 0)^T = 0$. This explains the fact that points with homogeneous coordinates $(x, y, 0)^T$ do not correspond to any finite point in R^2 and parallel lines meet at infinity [69].

Introduction of the concept of the points at infinity is essential because it serves to simplify the intersection properties of points and lines. So, one could state that, in the projective plane P^2, two distinct lines meet in a single point and two distinct points lie on a single line. This is not true in the standard Euclidean (rectilinear) geometry of R^2, in which parallel lines form a special case.

In the proposed work, users are asked to specify a rectangular area by the help of a user interface in order to extract planar facets from the images. Thus, having reliable information about the parallel lines over an image we can reveal the actual transformation that causes distortion for a plane.

2.1.1.2. Projective Transformation. A planar projective transformation is a linear transformation on homogenous 3-vectors represented by a non-singular 3x3 matrix in Equation 2.1 or more briefly **x' = Hx**. This is an invertible transformation from a projective plane to a projective plane that maps straight lines to straight lines. The most general transformation between the world and image plane under imaging by a perspective camera, is the projective transformation which is also called a "collineation", "homography", and "projectivity".

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{2.1}$$

We compute the projective transformation by defining point-to-point correspondences between user specified plane points and real rectilinear frontal corner points of a plane. Let the inhomogeneous coordinates of a pair of matching points **x** and **x'** in the world and image plane be (x, y) and (x', y') respectively. We use inhomogeneous coordinates here instead of the homogeneous coordinates of the points, because it is these inhomogeneous coordinates that are measured directly from the image and from the world plane. The projective transformation of Equation 2.1 can be written in inhomogeneous form as

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \qquad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{2.2}$$

which means each point correspondence generates two equations for the elements of **H**, which after multiplying out are

$$\begin{aligned} x'(h_{31}x + h_{32}y + h_{33}) &= h_{11}x + h_{12}y + h_{13} \\ y'(h_{31}x + h_{32}y + h_{33}) &= h_{21}x + h_{22}y + h_{23} \end{aligned} \tag{2.3}$$

Four point correspondences lead to eight such linear equations in the entries of **H**, which are sufficient to solve for **H** up to an insignificant multiplicative factor. The only restriction is that the four points must be in "general position", which means that no three points are collinear. The inverse of the transformation **H** computed in this way is then applied to the whole image to undo the effect of perspective distortion on the selected plane. The results are shown in Figure 2.1.

2.1.1.3. Image Warping. Images are warped by applying the inverse homography to each pixel in the target image. In order to automate the warping and ensure that the convex hull of the original image is correctly mapped into the rectangle of the target image, the intensities at source points in the original image are determined by bicubic interpolation.

During image rectification system also gathers an angle parameter for each image in order to use as a selection criterion. Details of this process is explained in Section 2.2.2.

2.2. Creating Texture Composite

This section begins with the description of the texture mapping concept and why we use multiple images for creating a texture composite (Section 2.2.1). Then, we will continue (Section 2.2.3) with the mathematical description of the Graph Cut Algorithm which is the main tool that we use for creating our texture composites. Two basic moves developed by Boykov et. al. [45] will be defined in the following section (Section 2.2.4). Then we will formulate the problem (Section 2.2.5) and go on with the details of one of these basic moves (Section 2.2.6).

2.2.1. Problem Description

Given a 3D building model and images taken from different sides and angles, to compute a texture map for each planar segment of the building model, we backproject

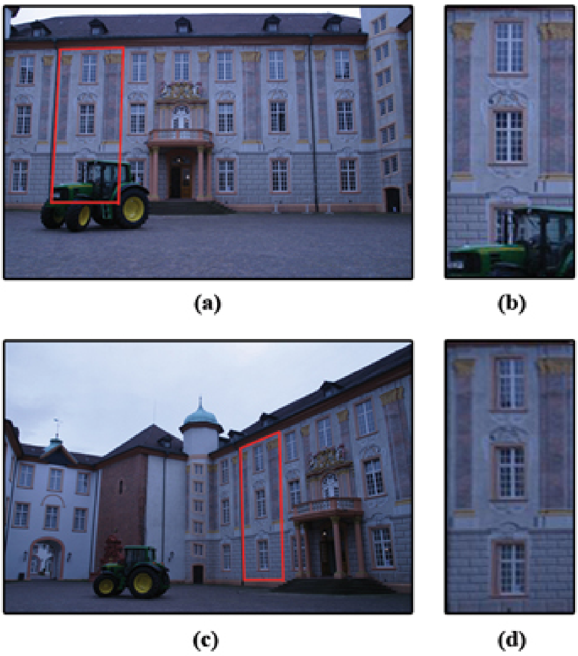


Figure 2.2. Rectification of a plane image from different views. Red lines represents the area of interest for rectification. a) Frontal image. b) Rectified area according to (a). c) Another view with a larger angle. d) Rectified area according to (c).

the selected and rectified (Section 2.1) views onto the plane. As the reconstructed plane is typically visible in multiple images, pixels from each of these backprojected images provide potential candidates for the texels in the target texture map. A simple way to obtain a texture map would be to blend together all the candidates for each texel, but mostly this produces noticeable ghosting artifacts if a misalignment or unmodeled geometric detail exists. Also losing the fine resolution existing in the images can be counted as another drawback.

Instead, to avoid these problems, we can choose the pixel values from the most frontal view for each texel independently. However this creates other artifacts in the form of noticeable seams in the texture map. Moreover, none of these approaches deal with the problem of partial occlusion of the modeled surface by foreground occluders.

So, as seen in Figure 2.2 (a,b), using the most frontal view generally satisfies the desired quality but comes across with the problem of occluding surfaces whereas choosing the side views with larger angles overcomes the occlusion problems but results in low quality textures after rectification as seen in Figure 2.2 (c,d).

Therefore, instead of trying to choose the best image and use it as the necessary texture image, we need a combination of the two approaches which will output a composite texture instead of a unique view. Thereby, each pixel in the resulting texture map will include the best image pixel for it among all views. However, this method generally yields undesired artifacts on the composite since the images contain negative effects of warping caused by the rectification phase and nature of the uncontrolled light conditions. An example of this situation is given in Figure 2.3. Obviously, the resulting composite in Figure 2.3 (b) is quite unacceptable because of the numerous seams. Moreover, looking at the labels generating the composite in Figure 2.3 (a), although most of the pixel values come from two main views, since they are spread over the image, they lack the desired quality. Thus, our preferred technique has to partition the image space between different images while generating image composites in order the minimize the unnecessary spreading. In other words, resulting texture map must include relatively larger areas of labelings while controlling the desired properties like the priority of the frontal views for a better quality.

The most recent and effective approach to deal with the above problem and to find optimal seams while retrieving texture composites from image sets is graph cut optimization. This technique applies texture map generation as a Markov Random Field (MRF) optimization problem, where a high quality seamless texture map is computed by minimizing a suitable energy functional consisting of data penalty terms and pairwise terms [45]. Lempitsky *et al* [70] used a similar technique for generating image-based texture maps and their underlying MRF was defined on a triangulated manifold mesh whereas we prefer to apply mapping over a texel grid of planar surfaces which is the case introduced in Sinha *et al* [12].

We denote the set of aligned images rectified to the target plane by $I_1, ..., I_n$. The graph cut estimates a label image f where the label at pixel p denoted by f_p indicates which image I_k should be used as the source for p in the target texture map. The energy functional we minimize is denoted by $E(f)$ where f is a particular label image.

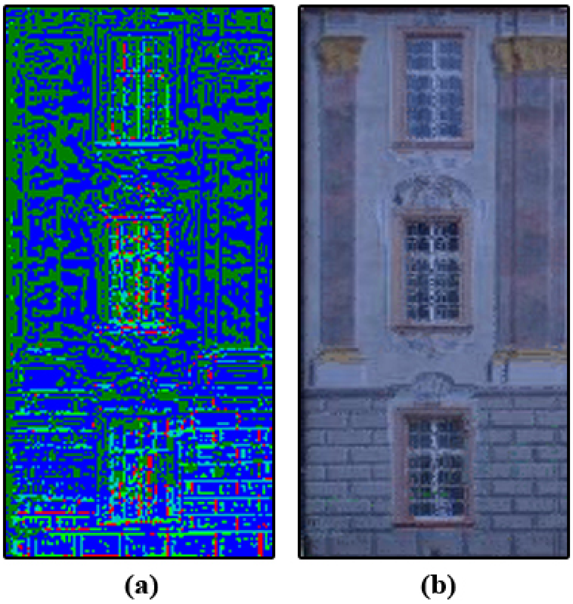


Figure 2.3. Creating texture composite by choosing best image for each pixel. a) Labeled image. Each label is represented with a different color. b) Image composite created according to the labels in (a).

Note that L is piecewise constant except at seams between adjacent pixels p and q, where $f_p \neq f_q$.

As we mentioned before, our energy functional E is the sum of a data penalty term summed over all pixels of the label image f and a pairwise interaction penalty term (sometimes called as smoothness term) summed over all pairs of neighboring pixels in f.

$$E(f) = E_{smooth}(f) + E_{data}(f) \tag{2.4}$$

Firstly, there are many image objectives that can be applied to the candidate images to use in the minimization function. The image objective at each pixel specifies a property that the user would like to see at each pixel in the composite. The image objective is computed independently at each pixel position p, based on the set of pixel values drawn from that same position p in each of the source images. We denote this set the *span* at each pixel.

The general image objectives that may be applied in a variety of applications include:

- Designated color: a specific desired color to either match or avoid.
- Minimum or maximum luminance: the darkest or lightest pixel in the span.
- Minimum or maximum contrast: the pixel from the span with the lowest or highest local contrast in the span.
- Minimum or maximum likelihood: the least or most common pixel value in the span.
- Eraser: the color most different from that of the current composite.
- Designated image: a specific source image in the stack.
- Minimum or maximum difference: the color least or most similar to the color at position p of a specific source image in the candidate set.

These objectives can be chosen according to the specific needs of the application used. Considering the concept of 3D reconstruction and texture mapping we will add following specific objectives to the list in order to obtain better composites.

- Preference for a frontal view: the pixel coming from the most frontal view with the angle θ, where θ is the angle between the plane normal and a particular camera. Automatic retrieval of this angle parameter is described in Section 2.2.2.
- Photo-consistency: the color most similar to the median color μ of the set of candidate pixels.

Here we will go through the details of the last two image objectives which will be used as the data term E_{data} in the original energy minimization function (Equation 2.4).

The data penalty term denoted by $D_p(f_p)$ stores the cost of assigning label f_p to pixel p while the interaction penalty term $V_{p,q}(f_p, f_q)$ stores the cost of assigning labels

f_p and f_q to neighboring pixels p and q in the label image.

$$E(f) = \sum_{\{p,q\}\in N} V_{p,q}(f_p, f_q) + \sum_{p\in P} D_p(f_p) \tag{2.5}$$

Two main objectives which defines our data penalty term can be formulated as follows:

$$\begin{aligned} D_1 &= 1 - cos^2(\theta), && (2.6)\\ D_2 &= \left|I_{f_p}(p) - \mu\right| && (2.7) \end{aligned}$$

The first objective in Equation 2.6 defines a high priority for the frontal images since θ is the angle between the plane normal and the camera. On the other hand, the second objective in Equation 2.7 introduces a color consistency objective which applies a high penalty for choosing a pixel whose color deviate from the median color μ.

It follows that our data penalty term becomes the weighted sum of the two data penalties.

$$E_{data} = D_p(f_p) = w_1(1 - cos^2(\theta)) + w_2\left|I_{f_p}(p) - \mu\right| \tag{2.8}$$

where w_1 and w_2 are two suitably chosen weights to balance the two objectives.

Interaction (smoothness) penalty term, on the other hand, is defined as the combination of matching colors and gradients across the seams. We can denote it as follows:

$$E_{smooth} = V_{p,q}(f_p, f_q) = \begin{cases} 0 & \text{if} \quad f_p = f_q \\ X + Y & \text{if} \quad o/w \end{cases} \tag{2.9}$$

$$\begin{aligned} X &= \left|I_{f_p}(p) - I_{f_q}(p)\right| + \left|I_{f_p}(q) - I_{f_q}(q)\right|, && (2.10)\\ Y &= \left|\nabla I_{f_p}(p) - \nabla I_{f_q}(p)\right| + \left|\nabla I_{f_p}(q) - \nabla I_{f_q}(q)\right| && (2.11) \end{aligned}$$

Using the sum of both color (Equation 2.10) and gradient (Equation 2.11) information to form the smoothness term generally gives better results. However one could also use this information separately for special cases as seen in Equation 2.12.

$$E_{smooth} = V_{p,q}(f_p, f_q) = \begin{cases} X & \text{if matching} & \text{colors} \\ Y & \text{if matching} & \text{gradients} \\ X + Y & \text{if matching} & \text{colors and gradients} \end{cases} \tag{2.12}$$

Boykov et. al. [45] applies graph cut optimization under two fairly general classes of interaction penalty V: metric and semimetric. V is called a metric on the space of labels L if it satisfies

$$V(\alpha, \beta) = 0 \;\Leftrightarrow\; \alpha = \beta, \tag{2.13}$$

$$V(\alpha, \beta) \;=\; V(\beta, \alpha) \geq 0, \tag{2.14}$$

$$V(\alpha, \beta) \;\leq\; V(\alpha, \gamma) + V(\gamma, \beta) \tag{2.15}$$

for any labels $\alpha, \beta, \gamma \in L$. If V satisfies only (2.13) and (2.14), it is called a semimetric.

Note that all of the smoothness objectives mentioned above are metrics since they always satisfy Equations (2.13), (2.14), and (2.15). [49] and [50] also add the edge information to the equation which may, in theory, enhance the quality of resulting image composite in terms of seams. The approach introduces the term $Z = H_{f_p}(p, q) + H_{f_q}(p, q)$ where $H_t(p, q)$ is the scalar edge potential between two neighboring pixels p and q of image t, computed using a Sobel filter. Edge potential is used with the color information by using the term X/Z which is a semi-metric since it does not always satisfy the triangle inequality in Equation 2.15. When this seam penalty is used, many of the theoretical guarantees of the "alpha expansion" algorithm are lost. It still gives acceptable results for some local, special cases; but while applying Poisson blending afterwards, edge term leads to artifacts through strong image edges. Therefore, this semi-metric is left beyond the scope of the approach used in this thesis.

2.2.2. Extracting Angle Criterion

During the elimination of the projective distortion we also have the opportunity to gather a parameter for each image in order to use as a selection criterion while creating the final texture image. Textures that will be created at the end of our system, may consist of pixel values coming from many different views. Thus, we need some criterions while deciding the dominance of each image in the final texture composites. One of these criterions is the closeness of the view to the frontal view of the same image. Such an information can be used to apply higher priorities for nearly frontal views in determining the final composites. Usage of frontal-view and many other criterions are considered in Section 2.2.1.

A good measure for defining how close a view to the perfect frontal view is the angle between plane normal and the vector coming from the viewing camera. Here the plane is the area that will be textured in 3D model and let θ be the defined angle. If we know θ for each view, then we can insert it into our system as one of the main selection measures. Yet, since we are not given the camera parameters, there is no direct way of calculating this angle. This is why we will estimate θ values for each view by using vanishing points.

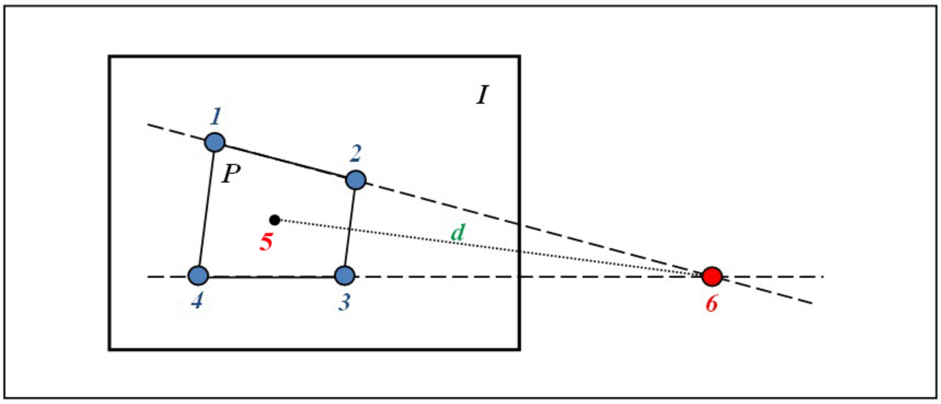


Figure 2.4. A view showing a plane under perspective distortion. Points 1 to 4 correspond to the corners of the plane and point 5 represents the intersection of the lines including two opposite edges of the plane in projective space.

As stated in Section 2.1.1.1, in general, parallel lines on a scene plane are not parallel in the image but instead converge to a finite point as seen in Figure 2.4. Because of the structure of inherent perspective the closeness of the view to the frontal view is increases with the distance d between the convergence point and the center of the plane. In other words, when distance d gets bigger then the amount of distortion increases and this means a larger viewing angle θ. In order to improve the conditioning of the equations and eliminate the effect of plane size over the distance d, we apply 2D normalization to the corner points so that their centroid will be at the origin.

This direct proportion between the distance d and angle θ allows us to estimate the viewing angle for each view by just making use of the points specified by the user. Since a plane is accepted as invisible in the current view for θ values over 90 degrees, distance values are mapped to an angle value θ between 0 and 90 degrees. Therefore, system estimates an angle measure θ for each view of the target plane in order to be used in the next step (Section 2.2.3) as a selection criterion.

2.2.3. Graph Cut Optimization

When we have a finite set of labels L, and we want to assign every pixel $p \in P$ to a label, in graph cut optimization, one seeks the labeling f that minimizes the energy

$$E(f) = E_{smooth}(f) + E_{data}(f) \tag{2.16}$$

Here, E_{smooth} measures the extent to which f is not piecewise smooth, while E_{data} measures the disagreement between f and the observed data. Many different energy functions have been proposed in the literature as stated in the previous part (Section 2.2.1). The form of E_{data} is typically

$$E_{data}(f) = \sum_{p \in P} D_p(f_p) \tag{2.17}$$

where D_p measures how well label f_p fits pixel p given the observed data. In image restoration, for example, $D_p(f_p)$ is normally, $(f_p - I_p)^2$ where I_p is the observed intensity of p.

The major difficulty with energy minimization lies in the enormous computational costs. Typically, these energy functions have many local minima (i.e., they are nonconvex). Worse still, the space of possible labelings has dimension $|P|$, which is many thousands.

The energy functions that are considered in this thesis arise in a variety of different contexts, including the Bayesian labeling of first-order Markov Random Fields. Here, the quality of pixel combinations are represented as a Markov Random Field and the problem is formulated as a minimum-cost graph-cut. Energies are considered of the form

$$E(f) = \sum_{\{p,q\} \in N} V_{p,q}(f_p, f_q) + \sum_{p \in P} D_p(f_p) \tag{2.18}$$

where N is the set of interacting pairs of pixels. Typically, N consists of adjacent pixels, but it can be arbitrary. We allow D_p to be nonnegative but otherwise arbitrary. According to the choice of E_{smooth}, only pairs of pixels interact directly. Note that each pair of pixels $\{p, q\}$ can have its own distinct penalty $V_{p,q}$.

2.2.4. Graph Cut Moves

Any labeling f can be uniquely represented by a partition of image pixels **P**$= \{P_l \,|\, l \in L\}$, where $P_l = \{p \in P \,|\, f_p = l\}$ is a subset of pixels which have been assigned the label l. Since there is an obvious one to one correspondence between labelings f and partitions **P**, we can use these notions interchangingly.

Given a pair of labels α, β, a move from a partition **P** (labeling f) to a new partition **P'** (labeling f') is called an *α-β-swap* if $P_l = P'_l$ for any label $l \neq \alpha, \beta$. This means that the only difference between **P** and **P'** is that some pixels that were labeled α in **P** are now labeled β in **P'**, and some pixels that were labeled β in **P** are now labeled α in **P'**. A special case of an *α-β-swap* is a move that gives the label α to some set of pixels previously labeled β.

Given a label α, a move from a partition **P** (labeling f) to a new partition **P'** (labeling f') is called an *α-expansion* if $P_\alpha \subset P'_\alpha$ and $P'_l \subset P_l$ for any label $l \neq \alpha$. In other words, an *α-expansion* move allows any set of image pixels to change their labels to α.

Given a labeling f, there is an exponential number of swap and expansion moves. Therefore, even checking for a local minimum requires exponential time if performed naively. In contrast, checking for a local minimum when only the standard moves are allowed is easy since there is only a linear number of standard moves given any labeling f.

When these moves are defined, it is easy to design variants of the "fastest descent" technique that can efficiently find the corresponding local minima. The algorithm for the expansion move is summarized as follows:

```
1. Start with an arbitrary labeling $f$
2. Set success := 0
3. For each label $\alpha \in L$
   (a) Find $\hat{f} = argminE(f')$ among $f'$ within one $\alpha$-expansion of $f$
   (b) If $E(\hat{f}) < E(f)$, set $f$:=$\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return $f$
```

Figure 2.5. Pseudocode for α-expansion algorithm

We will call a single execution of Steps (a), (b) an *iteration*, and an execution of Steps 2, 3, and 4 a *cycle*. In each cycle, the algorithm performs an iteration for every label (expansion algorithm) or for every pair of labels (swap algorithm), in a certain order that can be fixed or random. A cycle is successful if a strictly better labeling is found at any iteration. The algorithm terminates when a pass over all labels has occurred that fails to reduce the cost function. Thus, a cycle in the expansion algorithm takes $|L|$ iterations.

2.2.5. Graph Cuts

Let $G = \langle v, \varepsilon \rangle$ be a weighted graph with two distinguished vertices called the terminals. A *cut* $C \subset \varepsilon$ is a set of edges such that the terminals are separated in the induced graph $G(C) = \langle v, \varepsilon - C \rangle$. In addition, no proper subset of C separates the terminals in $G(C)$. The cost of the cut C, denoted $|C|$, equals the sum of its edge weights. The minimum cut problem is to find the cheapest cut among all cuts separating the terminals.

Normally, there are two types of edges in the graph: n-links and t-links. n-links connect pairs of neighboring pixels. Thus, they represent a neighborhood system in the image. Cost of n-links corresponds to a penalty for discontinuity between the pixels. These costs are usually derived from the pixel interaction term $V_{p,q}$ in energy (Equation 2.5). t-links connect pixels with terminals (labels). The cost of a t-link connecting a pixel and a terminal corresponds to a penalty for assigning the corresponding label to the pixel. This cost is normally derived from the data term D_p in the energy (Equation 2.5).

A *cut* C on a graph with two terminals is a partitioning of the nodes in the graph into two disjoint subsets A and B such that the source a is in A and the sink b is in B. Figure 2.6 shows one example of a cut. The *minimum cut* problem on a graph is to find a cut that has the minimum cost among all cuts. One of the fundamental results in this kind of optimization is that the minimum cut problem can also be solved by finding a maximum flow from the source a to the sink b. Loosely speaking, maximum flow is the maximum "amount of water" that can be sent from the source to the sink by interpreting graph edges as "pipes" with capacities equal to edge weights.

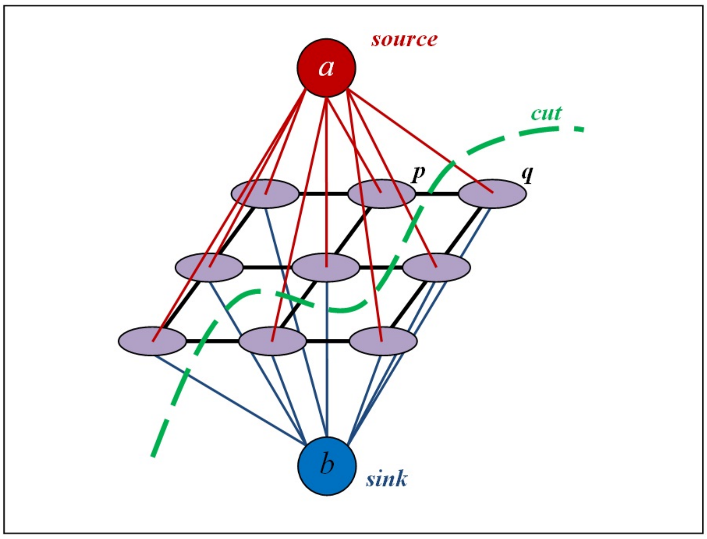


Figure 2.6. Example of a weighted graph and a sample cut.

It has been proven [71] that a maximum flow from a to b saturates a set of edges in the graph dividing the nodes into two disjoint parts A, B corresponding to a minimum cut. Therefore, min-cut and max-flow problems are equivalent and a "Duality" relationship exists between them. This is why these kind of problems are commonly named together as *min-cut/max-flow problems.*

Minimum cut is the preferred strategy in this work to optimize our energy function. For simplification, a natural labeling will be defined for each cut C in the following section (Section 2.2.6) and this labeling will be directly used in cost calculations.

2.2.6. Finding the Optimal Expansion Move

Given an input labeling f (partition $\mathbf{P}$) and a label α, we would like to find a labeling f' that minimizes E over all labelings within one α-*expansion* of f. Here, we describe the technique that solves the problem assuming that (each) V is a metric and, thus, satisfies the triangle inequality (2.15). The technique is based on computing a labeling corresponding to a minimum cut on a graph $G_\alpha = \langle \upsilon_\alpha, \varepsilon_\alpha \rangle$. The structure of this graph is determined by the current partition $\mathbf{P}$ and by the label α. So, the graph dynamically changes after each iteration.

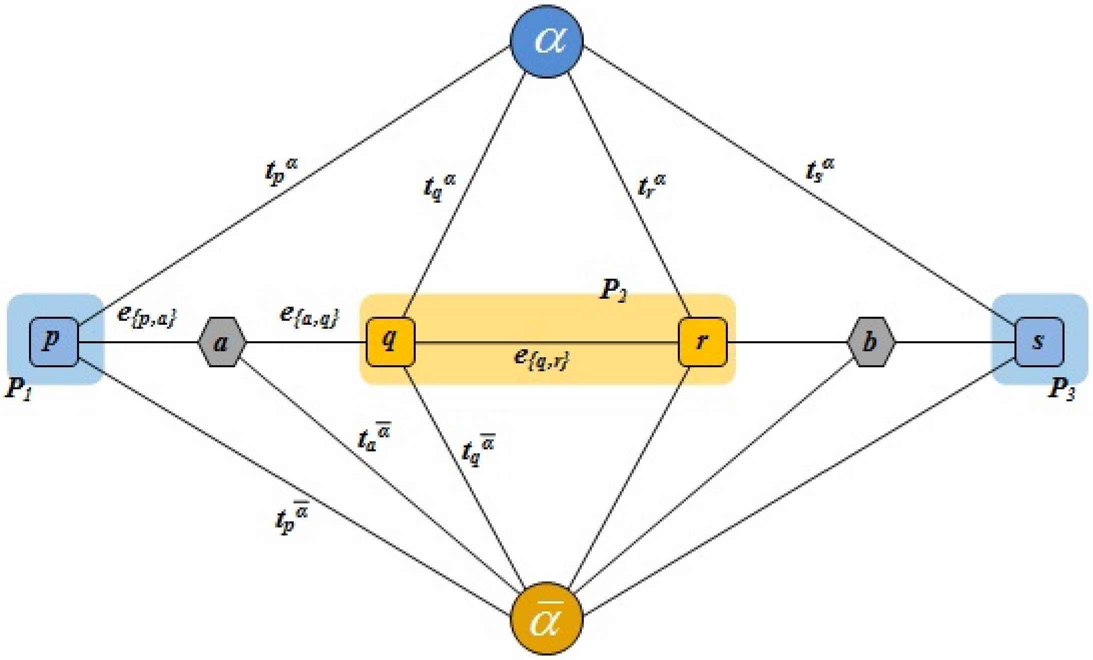


Figure 2.7. An example of G_α for a 1D image.

Figure 2.7 shows the case of a 1D image. Here, the set of pixels in the image is $P = \{p, q, r, s\}$ and the current partition for the figure is $\mathbf{P} = \{P_1, P_2, P_3\}$, where$P_1 = \{p\}, P_2 = \{q, r\}$,and $P_3 = \{s\}$. For each pair of neighboring pixels $\{p, q\} \in N$ separated in the current partition (i.e., such that $f_p \neq f_q$), we create an *auxiliary node* $a_{\{p,q\}}$. Auxiliary nodes which are shown as $a = a_{\{p,q\}}$ and $b = a_{\{r,s\}}$ in the Figure 2.7, are introduced at the boundaries between partition sets P_l for $l \in L$. Thus, the set of vertices is

$$V_\alpha = \left\{ \alpha, \bar{\alpha}, \bigcup_{\substack{\{p,q\} \in N \\ f_p \neq f_q}} a_{\{p,q\}} \right\} \tag{2.19}$$

Each pixel $p \in P$ is connected to the terminals α and $\bar{\alpha}$ by t-links t_p^{α} and $t_p^{\bar{\alpha}}$, respectively. Each pair of neighboring pixels $\{p, q\} \in N$ which are not separated by the partition $\mathbf{P}$ (i.e., such that $f_p = f_q$) is connected by an n-link $e_{\{p,q\}}$. For each pair of neighboring pixels $\{p, q\} \in N$ such that $f_p \neq f_q$,we create a triplet of edges $\varepsilon_{\{p,q\}} = \{e_{\{p,a\}}, e_{\{a,q\}}, t_a^{\bar{\alpha}}\}$, where $a = a_{\{p,q\}}$ is the corresponding auxiliary node. The edges $e_{\{p,a\}}$ and $e_{\{a,q\}}$ connect

Table 2.1. Weights assigned to the edges.

Edge	**Weight**	**for**
$t_p^{\bar{\alpha}}$	∞	$p \in P_\alpha$
$t_p^{\bar{\alpha}}$	$D_p(f_p)$	$p \notin P_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in P$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in N, f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	$\{p,q\} \in N, f_p \neq f_q$
$t_a^{\bar{\alpha}}$	$V(f_p, f_q)$	$\{p,q\} \in N, f_p \neq f_q$
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in N, f_p = f_q$

pixels p and q to $a_{\{p,q\}}$ and the t-link $t_a^{\bar{\alpha}}$ connects the auxiliary node $a_{\{p,q\}}$ to the terminal $\bar{\alpha}$. So, we can write the set of all edges as

$$\varepsilon_\alpha = \left\{ \bigcup_{p \in P} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\} \in N \\ f_p \neq f_q}} \varepsilon_{\{p,q\}}, \bigcup_{\substack{\{p,q\} \in N \\ f_p = f_q}} e_{\{p,q\}} \right\} \tag{2.20}$$

The weights assigned to these edges can be determined as in the Table 2.1.

Any cut C on G_α must include exactly one t-link for any pixel $p \in P$. This defines a natural labeling f^C corresponding to a cut C on G_α. Formally,

$$f^C = \begin{cases} \alpha & \text{if} \quad t_p^\alpha \in C \\ f_p & \text{if} \quad t_p^{\bar{\alpha}} \in C \end{cases} \qquad \forall p \in P \tag{2.21}$$

In other words, a pixel p is assigned label α if the cut C separates p from the terminal α, while p is assigned its old label f_p if C separates p from $\bar{\alpha}$. Note that, for $p \notin P_\alpha$, the terminal $\bar{\alpha}$ represents labels assigned to pixels in the initial labeling f. This follows that

Lemma 2.1 *A labeling f^C corresponding to a cut C on G_α is one α-expansion away* *from the initial labeling f.*

It is obvious that a cut C includes an n-link $e_{\{p,q\}}$ between neighboring pixels $\{p,q\} \in N$ such that $f_p = f_q$ if and only if C leaves the pixels p and q connected to different terminals. Formally,

Property 2.1 *For any cut C and for any n-link $e_{\{p,q\}}$:*

(a) *If* $t_p^{\alpha}, t_q^{\alpha} \in C$ *then* $e_{\{p,q\}} \notin C$
(b) *If* $t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in C$ *then* $e_{\{p,q\}} \notin C$
(c) *If* $t_p^{\bar{\alpha}}, t_q^{\alpha} \in C$ *then* $e_{\{p,q\}} \in C$
(d) *If* $t_p^{\alpha}, t_q^{\bar{\alpha}} \in C$ *then* $e_{\{p,q\}} \in C$

Properties 2.1 (a) and 2.1 (b) follow from the requirement that no proper subset of C should separate the terminals. Properties 2.1 (c) and 2.1 (d) also use the fact that a cut has to separate the terminals. These properties are illustrated in Figure 2.8. The next lemma is a consequence of Property 2.1 and (2.21).

Lemma 2.2 *For any cut C and for any n-link $e_{\{p,q\}}$*

$$\left| C \cap e_{\{p,q\}} \right| = V(f_p^C, f_q^C). \tag{2.22}$$

Now, consider the set of edges $\varepsilon_{\{p,q\}}$ corresponding to a pair of neighboring pixels $\{p,q\} \in N$ such that $f_p \neq f_q$. In this case, there are several different ways to cut these edges even when the pair of severed t-links at p and q is fixed. However, a minimum cut C on G_α is guaranteed to include the edges in $\varepsilon_{\{p,q\}}$ depending on what t-links are cut at the pixels p and q. The rule for this case is described in Property 2.2. Assume that $a = a_{\{p,q\}}$ is an auxiliary node between the corresponding pair of neighboring pixels.

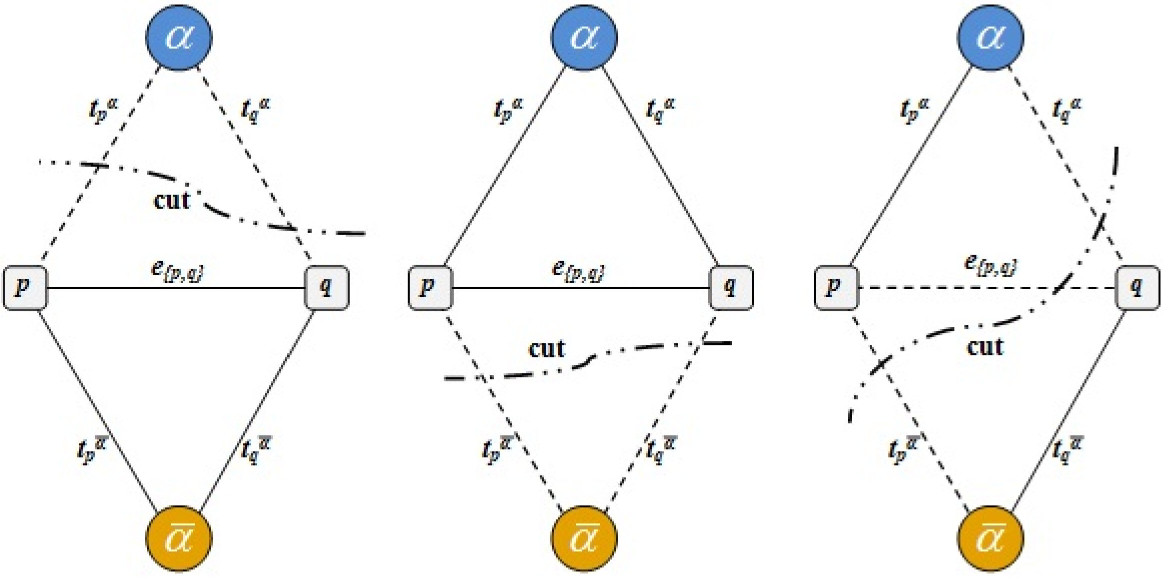


Figure 2.8. Properties of a cut C on G_α for two pixels $\{p, q\} \in N$ connected by an n-link $e_{\{p,q\}}$. Dotted lines show the edges cut by C and solid lines show the edges remaining in the induced graph.

Property 2.2 *If $\{p, q\} \in N$ and $f_p \neq f_q$, then a minimum cut C on G_α satisfies:*

(a) *If* $t_p^\alpha, t_q^\alpha \in C$ *then* $C \cap \varepsilon_{\{p,q\}} = \emptyset$

(b) *If* $t_p^{\bar\alpha}, t_q^{\bar\alpha} \in C$ *then* $C \cap \varepsilon_{\{p,q\}} = t_a^{\bar\alpha}$

(c) *If* $t_p^{\bar\alpha}, t_q^\alpha \in C$ *then* $C \cap \varepsilon_{\{p,q\}} = e_{\{p,a\}}$

(d) *If* $t_p^\alpha, t_q^{\bar\alpha} \in C$ *then* $C \cap \varepsilon_{\{p,q\}} = e_{\{a,q\}}$

Property 2.2 (a) results from the fact that no subset of C is a cut. The others follow from the minimality of $|C|$ and the fact that $\left|e_{\{p,a\}}\right|, \left|e_{\{a,q\}}\right|$ and $|t_a^{\bar\alpha}|$ satisfy the triangle inequality so that cutting any one of them is cheaper than cutting the other two together. These properties are illustrated in Figure 2.9.

Lemma 2.3 *If $\{p, q\} \in N$ and $f_p \neq f_q$, then the minimum cut C on G_α satisfies*

$$\left|C \cap \varepsilon_{\{p,q\}}\right| = V(f_p^C, f_q^C). \tag{2.23}$$

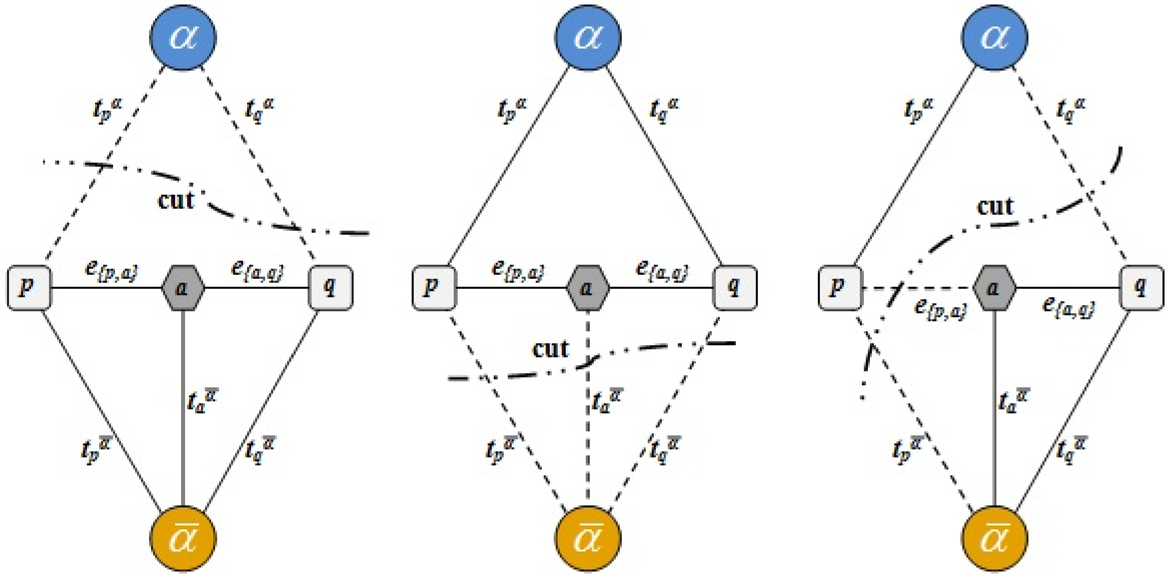


Figure 2.9. Properties of a minimum cut C on G_α for two pixels $\{p, q\} \in N$ such that $f_p \neq f_q$. Dotted lines show the edges cut by C and solid lines show the edges remaining in the induced graph.

Equation 2.23 comes from Property 2.2, Equation 2.21 and the edge weights. For example, if $t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in C$, then $\left|C \cap \varepsilon_{\{p,q\}}\right| = |t_a^{\bar{\alpha}}| = V(f_p, f_q)$, since Equation 2.21 implies that $f_p^C = f_p$ end $f_q^C = f_q$.

So far, we constructed the necessary mathematical background for finding the optimal graph cut. To sum up, let G_α be constructed as above. Given f and α we can say that, there is a one to one correspondence between elementary cuts on G_α and labelings within one α-*expansion* of f. So, for any elementary cut C, we can calculate $|C| = E(f^C)$ as follows.

We already show that an elementary cut C can be determined by the corresponding labeling f^C. The label f_p^C at the pixel p determines which of the t-links to p is in C. Property 2.1 shows which n-links $e_{\{p,q\}}$ between pairs of neighboring pixels $\{p, q\}$ such that $f_p = f_q$ are included. And, Property 2.2 determines which of the links in $\varepsilon_{\{p,q\}}$ corresponding to $\{p, q\} \in N$ such that $f_p \neq f_q$ should be in the cut. Putting

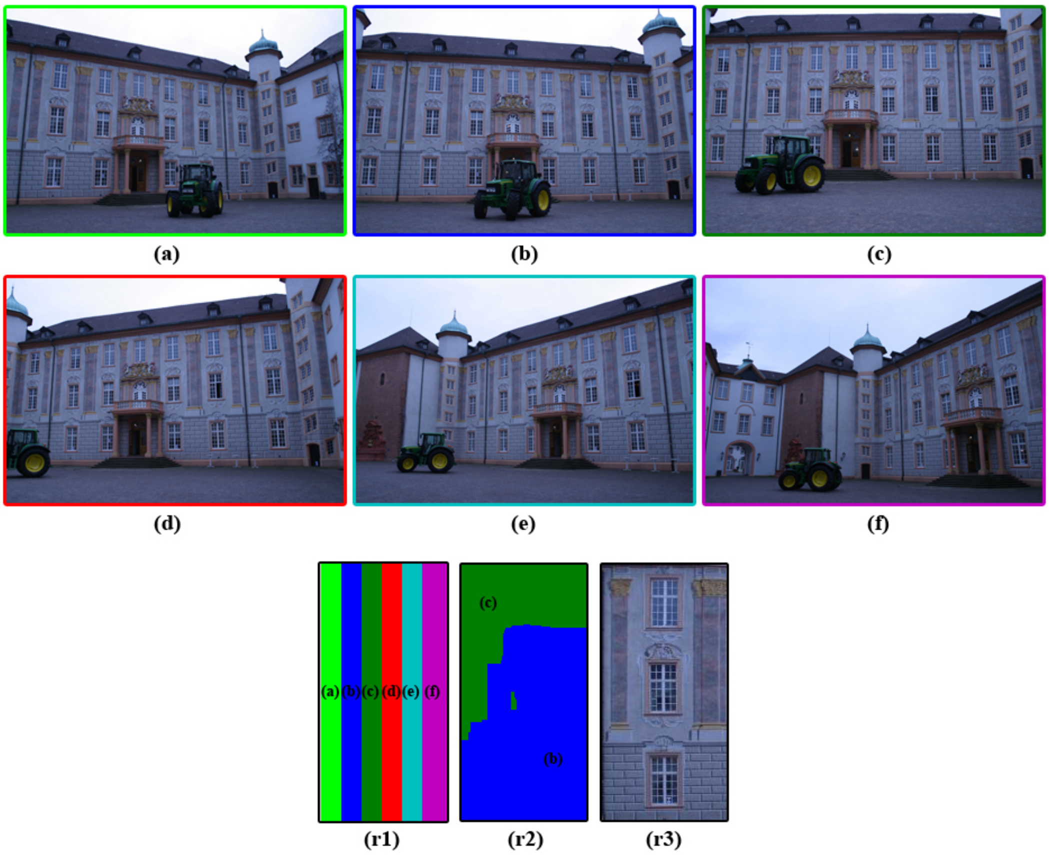


Figure 2.10. An example of a texture composite built up using α-expansion move. (a-f) Candidate images, (r1) Initial labeling, (r2) Final labeling, (r3) Image composite according to the final labeling.

these together, we can define the cost of an elementary cut C as

$$|C| = \sum_{p \in P} \left| C \cap \{t_p^{\alpha}, t_p^{\bar{\alpha}}\} \right| + \sum_{\substack{\{p,q\} \in N \\ f_p = f_q}} \left| C \cap e_{\{p,q\}} \right| + \sum_{\substack{\{p,q\} \in N \\ f_p \neq f_q}} \left| C \cap \varepsilon_{\{p,q\}} \right| \tag{2.24}$$

For any pixel $p \in P$, we have $\left| C \cap \{t_p^{\alpha}, t_p^{\bar{\alpha}}\} \right| = D_p(f_p^C)$. Using Lemmas 2.2 and 2.3 we can rewrite the total cost of an elementary cut C as

$$|C| = \sum_{p \in P} D_p(f_p^C) + \sum_{\{p,q\} \in N} V(f_p^C, f_q^C) = E(f^C) \tag{2.25}$$

As a result we can say that the lowest energy labeling within a single α-*expansion* move from f is $\hat{f} = f^C$, where C is the minimum cut on G_{α}.

Figure 2.10 shows a sample graph cut optimization process using α-*expansion* as the necessary move operator. First 6 images (a-f) represent different views of the same scene from different angles and form a candidate set for the texture composite. Along with the view information (the angle between the plane normal of the target wall and the related camera) gathered during rectification (Section 2.2.2), each candidate is labeled with a different color and a letter. Proposed system, firstly, calculates the data penalty terms according to the objectives explained in Section 2.2.1. Then an arbitrary initial labeling (Figure 2.10 (r1)) is defined to begin with, in the first Graph Cut cycle. Before each iteration smoothness (interaction) penalty terms are calculated according to the labeling at that moment. Consequently, Graph Cut algorithm yields a final minimized labeling (Figure 2.10 (r2)) and forms the texture composite (Figure 2.10 (r3)) according to this labeling. After this point system requires the decision of the user in order to use the final composite as texture or continue with further refinements explained in Section 2.3.

2.3. Texture Refinement

Since we deal with real images taken with a standard digital camera and uncontrolled lighting for many applications the source images are too dissimilar for a graph-cut alone to result in visually seamless composites. If the graph-cut optimization cannot find ideal seams, artifacts may still exist. Therefore a refinement phase becomes necessary for building seamless texture maps for the targeted planar surface.

Figure 2.11 demonstrates a graph cut process when there are too few images available and these candidate images are quite different in terms of luminance. This kind of situations generally brings out remarkable seams in the resulting composites as in Figure (b).

In these cases, it is useful to view the input images as sources of color gradients rather than sources of color. Using the same graph cut labeling, we copy color gradients to form a composite vector field. We then calculate a color composite whose gradients best match this vector field. Doing so allows us to smooth out color differ-

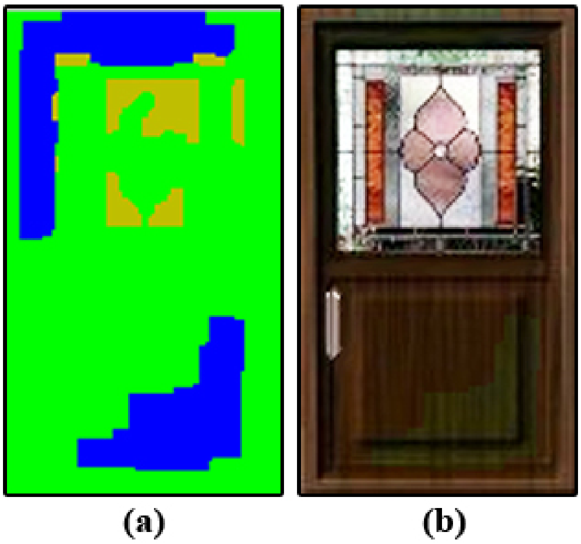


Figure 2.11. A sample graph cut process that results with a texture composite with seams. a) Final labeling after graph cut optimization. b) Final image according to (a).

ences between juxtaposed image regions. This process is commonly called as *Poisson blending* or with a more general name as *gradient-domain fusion.* Many image editing tools like selection editing, texture flattening etc. make use of this technique in order to gather seamless results. Here we will use Poisson blending for applying seamless cloning between different views with different labels in order to gather seamless texture composites.

Firstly, we will give the basics of the gradient-domain fusion approach (Section 2.3.1). After formulating and defining guidance vector (Section 2.3.2) we will continue with the discretization of the concept (Section 2.3.3) in order to apply to the two-dimensional image world.

2.3.1. Gradient Domain Fusion

The main idea of the approach is the Poisson partial differential equation with Dirichlet boundary conditions which specifies the Laplacian of an unknown function over the domain of interest, along with the unknown function values over the boundary of the domain. Tools created with this idea are used in many different areas like seamless cloning, selection editing, and seamless tilling [50, 54]. Seamless cloning best fits the needs of our seam problems occurred in composite texture images since these textures are composed of partitions each coming from an image in the candidate list. If we build up a source-destination relationship between this image partitions then we will apply seamless cloning for each partition.

The seamless cloning tool basically defined as the transfer process of an image region to another image or another area of the same image. The seams formed because of this process are overcame by a fusion process in gradient domain. Figure 2.12 represents a seamless cloning example in which a male profile is replaced with the face of the famous painting Mona Lisa. After defining the area of cloning, this area is pasted over destination image (Figure 2.12 (e)). Then a guided interpolation process, which will be detailed in Section 2.3.2, takes place in order to create a new seamless composite by determining color flow information from the source image and color information from the destination image. Loosely speaking, this technique stays loyal to the amount of color transitions between pixels in source image while ensures the compliance of source and destination boundaries. Thus the resulting clone does not include undesirable seams (Figure 2.12 (f)) compared to the direct cloning (Figure 2.12 (e)).

Cloning area is behaved as an unknown function over some domain. Given this domain, and the boundary conditions of the unknown function, the Poisson equation can be solved numerically to achieve seamless filling of that domain. This can be replicated independently in each of the channels of a color image. Solving the Poisson equation also has an alternative interpretation as a minimization problem: it computes the function whose gradient is the closest to some prescribed vector field - the guidance vector field - under given boundary conditions. This guidance vector field is determined using the source image.

Actually, unless the guidance vector field is conservative, no image exists whose gradient exactly matches the input. Instead, a best-fit image in a least-squares sense can be calculated by solving a discretization of the Poisson equation. Sections 2.3.2 and 2.3.3 include necessary mathematical formulation to build up the system so as to

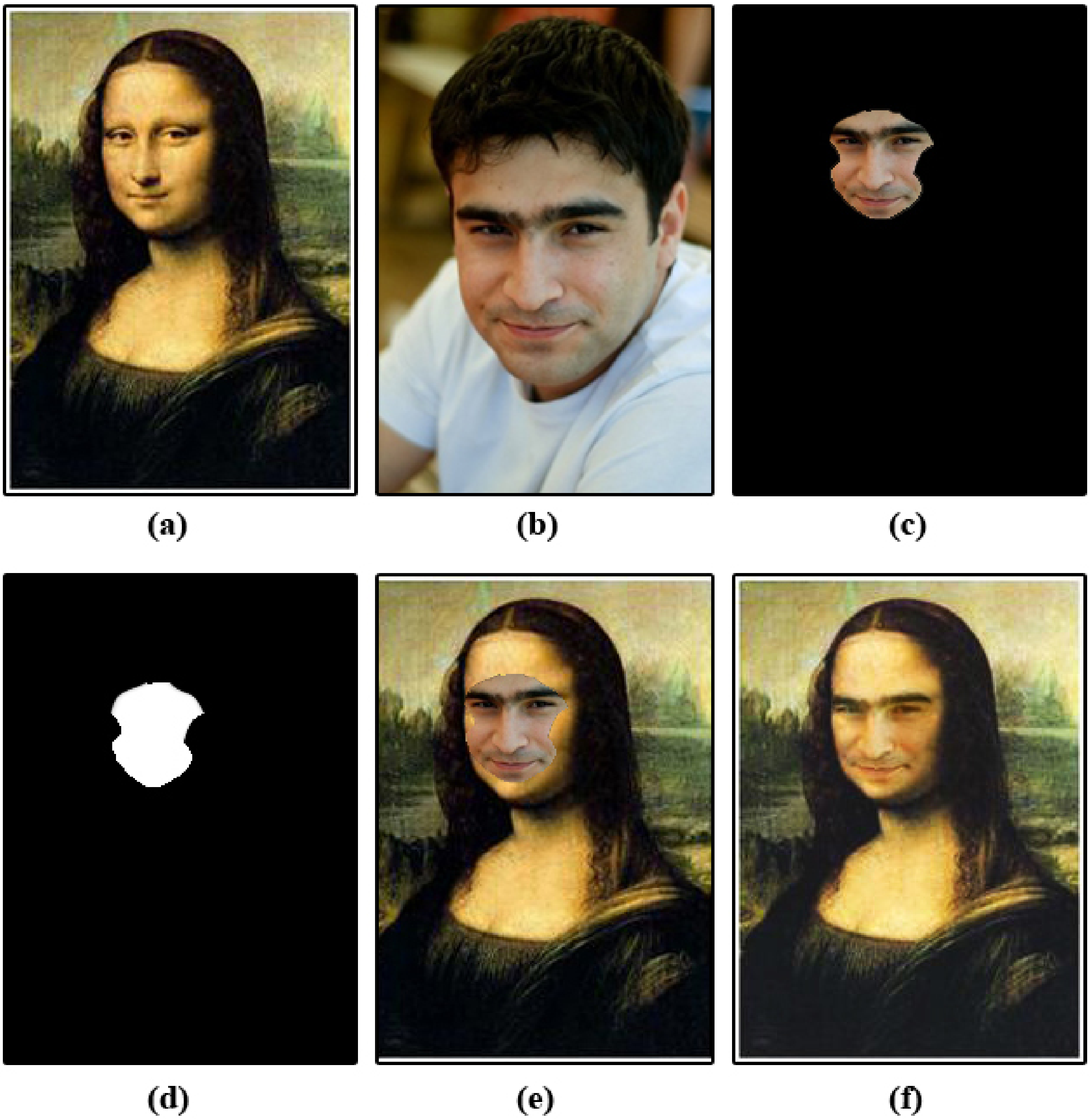


Figure 2.12. Example of a seamless cloning process. (a) Destination image, (b) Source image, (c) Area that will be transferred to the destination image, (d) Clone mask, (e) Destination image with the selected region pasted over, (f) Final image after Poisson seamless cloning.

seamlessly fill in the targeted cloning area.

2.3.2. Guided Interpolation

In this section, we detail image interpolation using a guidance vector field. As it is enough to solve the interpolation problem for each color component separately, we consider only scalar image functions. Figure 2.13 illustrates the notations: let S, a closed subset of R^2, be the image definition domain, and let Ω be a closed subset of S with boundary $\partial\Omega$. Let f^* be a known scalar function defined over S minus the interior of Ω and let f be an unknown scalar function defined over the interior of Ω. Finally, let $\mathbf{v}$ be a vector field defined over Ω.

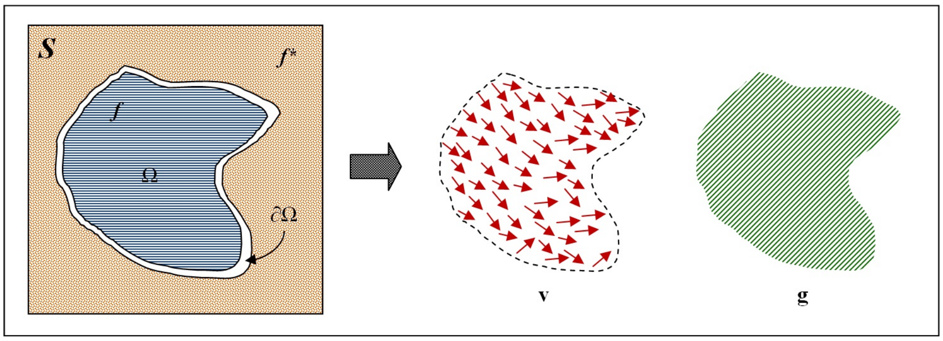


Figure 2.13. Guided Interpolation notation. Unknown function f interpolates in domain Ω the destination function f^*, under guidance of vector field $\mathbf{v}$, which might be or not the gradient field of a source function g.

The simplest interpolant f of f^* over Ω is the membrane interpolant defined as the solution of the minimization problem:

$$\min_f \iint_\Omega |\nabla f|^2 \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.26}$$

where $\nabla. = \left[\frac{\partial.}{\partial x}, \frac{\partial.}{\partial y}\right]$ is the gradient operator. The minimizer must satisfy the associated Euler-Lagrange equation

$$\Delta f = 0 \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.27}$$

where $\Delta. = \left[\frac{\partial^2 .}{\partial x^2}, \frac{\partial^2 .}{\partial y^2}\right]$ is the Laplacian operator. Equation 2.27 is a Laplace equation with Dirichlet boundary conditions that correspond to the constraint that the values at the edge of the region must match the destination image's value there.

For image editing applications, this simple method produces an unsatisfactory, blurred interpolant, and this can be overcome in a variety of ways. One is to use a more complex differential equation as in the "inpainting" technique of [61]. However, the route proposed here is to modify the problem (Equation 2.26) by introducing further constraints in the form of a guidance field as explained below.

A guidance field is a vector field $\mathbf{v}$ used in an extended version of the minimization problem (Equation 2.26) and simply assists the interpolation process in a desired way. Therefore, Equation 2.26 becomes

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \text{with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.28}$$

whose solution is the unique solution of the following Poisson equation with Dirichlet boundary conditions:

$$\Delta f = div\mathbf{v} \text{ over } \Omega \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega} \tag{2.29}$$

where $div\mathbf{v} = \left[\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}\right]$ is the divergence of $\mathbf{v} = (u, v)$. Here u and v stands for the components along x and y directions respectively.

When the guidance field $\mathbf{v}$ is conservative, i.e., it is the gradient of some function g, a helpful alternative way of understanding what Poisson interpolation does is to define the correction function Ω such that $\tilde{f}$ on $f = g+\tilde{f}$. The Poisson equation (Equation

2.29) then becomes the following Laplace equation with boundary conditions:

$$\Delta\tilde{f} = 0 \text{ over } \Omega \text{ with } \tilde{f}\Big|_{\partial\Omega} = (f^* - g)|_{\partial\Omega} \tag{2.30}$$

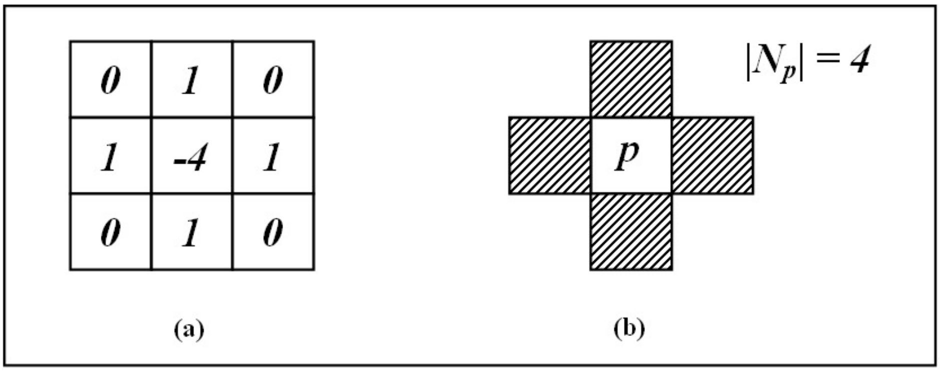
Therefore, inside Ω, the additive correction $\tilde{f}$ is a membrane interpolant of the mismatch $(f^* - g)$ between the source and the destination along the boundary $\partial\Omega$.

This is the fundamental machinery of Poisson editing of color images. When our application area, seamless cloning, is considered this machinery follows up the following mapping of notations: S represents the domain of the destination in cloning process whereas Ω stands for area of interest. Boundaries of the cloning area is determined by $\partial\Omega$ and let f^* be the destination image minus the cloning area Ω. Consequently, f represents the unknown pixel values that are searched under the guidance of vector field $\mathbf{v}$ which basically consists of the gradient of the source image in Ω. Nextly (Section 2.3.3), we will discretize our system in order to apply for 2D image coordinates in this notation.

2.3.3. Discrete Poisson Solver

The variational problem (Equation 2.28), and the associated Poisson equation with Dirichlet boundary conditions (Equation 2.29), can be discretized and solved in a number of ways. For discrete images the problem can be discretized naturally using the underlying discrete pixel grid. Since an RGB image corresponds to three 2D functions we will treat color channels separately and solve the following discrete formulation for each color independently.

Without loss of generality, we will keep the same notations for the continuous objects and their discrete counterparts: S, Ω now become finite point sets defined on an infinite discrete grid. Note that S can include all the pixels of an image or only a subset of them. For each pixel p in S, let N_p be the set of its 4-connected neighbors which are in S, and let $\langle p, q\rangle$ denote a pixel pair such that $q \in N_p$. The boundary of Ω is now $\partial\Omega = \{p \in S\ \Omega : N_p \cap \Omega \neq \emptyset\}$. Let f_p be the value of f at p. The task is to compute the set of intensities $f|_\Omega = \{f_p, p \in \Omega\}$.



0	1	0
1	-4	1
0	1	0

Figure 2.14. (a) A laplace operator, (b) Neighborhood of pixel p.

The function minimized in Equation 2.26 can be written as $\nabla f(x,y) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$. This represents partial derivatives of a multivariate function and we can refer this equation as the image gradient. Equation 2.31 shows how finite differences are computed in order to represent image gradient in a discrete manner.

$$\frac{\partial f}{\partial x} \approx f(x+1,y) - f(x,y), \qquad \frac{\partial f}{\partial y} \approx f(x,y+1) - f(x,y) \tag{2.31}$$

where x and y specifies the column and row values of any pixel p. The laplace operator, on the other hand, used in Poisson equation (Equation 2.29) can be shown as $\Delta f = \nabla^2 f = \left[\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2}\right]$ and this can be specified with the following equation according to the well-known 2D laplace operator in Figure 2.14 (a).

$$\Delta f = \nabla^2 f \approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{2.32}$$

Turning back to the context of seamless cloning, guidance vector field $\mathbf{v}$ can be thought as ∇g where g is the function representing source image and for all $\langle p,q \rangle$, we can write $v_{pq} = g_p - g_q$. Therefore, Equation 2.29 becomes

$$\Delta f = \Delta g \text{ over } \Omega, \text{ with } f|_{\partial\Omega} = f^*|_{\partial\Omega}\, N_p v_{pq} \tag{2.33}$$

However, generally speaking, for Dirichlet boundary conditions defined on a boundary of arbitrary shape, it is best to discretize the variational problem (Equation 2.28) directly, rather than the Poisson equation (Equation 2.29). Replacing continues functions with discrete image context we can read Equation 2.28 as finding image f such that its gradients as similar as possible to given gradient field $\mathbf{v}$ and satisfies the boundary conditions. The finite difference discretization of this equation yields the following discrete, quadratic optimization problem:

$$\min_{f|_\Omega} \sum_{\langle p,q\rangle \cap \Omega \neq \emptyset} (f_p - f_q - v_{pq})^2, \text{ with } f_p = f_p^*, \text{ for all } p \in \partial\Omega \tag{2.34}$$

where v_{pq} is the projection of $v\left(\frac{p+q}{2}\right)$ on the oriented edge $[p, q]$, i.e., $v_{pq} = v\left(\frac{p+q}{2}\right)\vec{pq}$. Its solution satisfies the following simultaneous linear equations:

$$\text{for all } p \in \Omega, |N_p|\, f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p \cap \partial\Omega} f_q^* + \sum_{q \in N_p} v_{pq} \tag{2.35}$$

Here, N_p defines the neighborhood given in Figure 2.14 (b). When Ω contains pixels on the border of S, which happens for instance when Ω extends to the edge of the pixel grid, these pixels have a truncated neighborhood such that $|N_p| < 4$. Note that for pixels p interior to Ω, that is, $N_p \cap \Omega$, there are no boundary terms in the right hand side of Equation 2.35, which reads:

$$|N_p|\, f_p - \sum_{q \in N_p \cap \Omega} f_q = \sum_{q \in N_p} v_{pq} \tag{2.36}$$

Equations 2.35 form a classical, sparse (banded), symmetric, positive-definite system. Because of the arbitrary shape of boundary $\partial\Omega$, we have been computed the results by using Gauss-Seidel iteration, one of the well-known iterative solvers.

2.3.4. Blending Tools in Application

In our texture mapping system we implemented two main tools by using Poisson blending: Local Blending and Smart Blending. Local blending is applied over a rectangular area and, as usual gives good results in removing rectangular seams. Moreover, this technique requires user interaction for specifying area of blending. Smart Blending, on the other hand, requires no user interaction since it defines the areas of interest automatically by looking at the final partitioning of candidate images over the final composite of labeling.

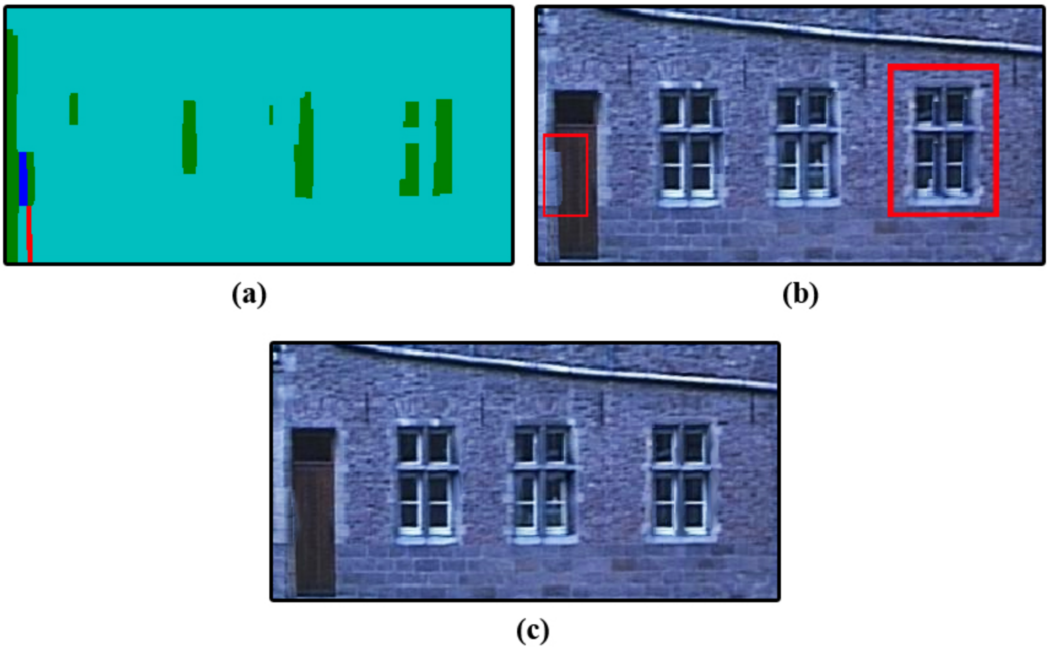


Figure 2.15. An example for smart blending. a) Final labeling after graph cut optimization. b) Image according to (a). c) Final image after smart blending.

Figure 2.15 (a) demonstrates a labeling for a final texture composite created after a Graph Cut iteration. In most cases, one of the candidate images dominates the final image since it satisfies a high priority objective like being the most frontal view. The first thing we do in Smart Blending is to find this dominant label in the composite image. Then we create a set of blend targets from the partitions lying over the dominant label. The images corresponding to each partition represents source images while the image corresponding to the dominant label forms the destination image. Thus, the system traverses each partition and applies seamless cloning between the sources and the destination. Figure 2.15 (c) exemplifies the resulting texture image after smart blending.

The detailed usage of both blending tools is given in Section 3.2.

3. TEXTURE MAPPING APPLICATION

Up until now, mathematical descriptions of the approaches we applied are issued in order to propose a stable solution to the texture mapping problem. In this chapter we will describe the tools developed in order to provide a user interface which simplifies the texture mapping process.

3.1. User Interface Properties

Sketch-based modeling interfaces are the main inspiration for our system [11, 12, 72]. These interfaces allow users to quickly create 3D models from simple 2D drawings and gestures. Yet most of these applications provide 3D reconstruction interface and tools but in this application we are mainly focusing on texture refinement and mapping of a pre-constructed 3D model.

Our system allows the user to visualize, edit and refine the textures using an interactive interface. Photographs are rectified into planar surfaces, providing visual feedback during geometric editing. This allows users to accelerate tasks by simply accepting or rejecting the images as candidates.

Implemented user interface gives the opportunity to select each planar surface by visually specifying corner points. Additionally, an optional automatic corner detector is provided to allow ease of selection especially for images with fuzzy details. Snapping the user specified points to the corners detected, we provide more accurate planar images by eliminating possible user oriented click errors.

In order to texture the model, our system generates texture maps using graph cut optimization and Poisson blending to compute seamless texture composites by combining patches from multiple input photographs. During this step, the user can exclude undesired pixels by using exclusion tool so that the system creates new composites by using different candidates to fill the specified areas. Unsatisfactory results of graph cut operation can be eliminated by a local or global blending to remove seams. Also a smart blending tool is provided in order to minimize the interaction level.

Two other useful tools are implemented within the application. One of them is the exclusion tool which lets users to specify the undesired areas of the final texture. Therefore system operates a new graph cut iteration by excluding the specified areas (applying high penalties for the labels in the specified areas). While using this tool users can also decide the size of the exclusion brush in order to specify the undesired area more accurately. The second feature implemented gives the opportunity to select a predefined texture as the texture of another plane. This tool is useful especially for buildings which include repeating planes like windows or same wall structures. A flip property comes with this feature in order to texture symmetric structures quickly.

3.2. Interface Usage

The application has two modes: Selection Mode and Refinement Mode (Figure 3.1).

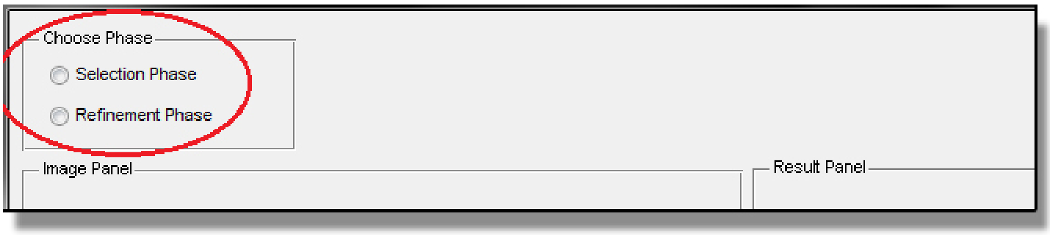
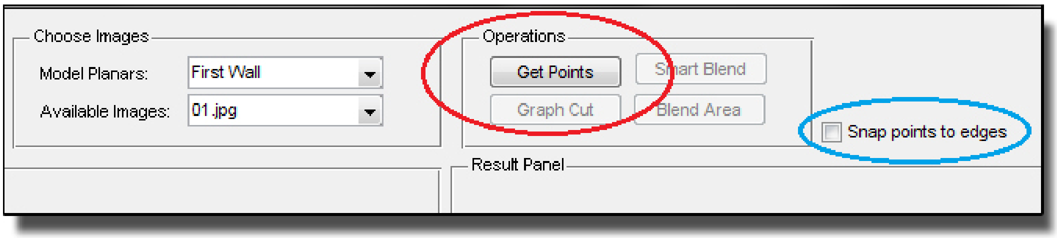


Figure 3.1. Menu for choosing the mode of application.

3.2.1. Selection Mode

Selection mode allows users to view candidate images for a real world object. In order to extract the planar textures “Get Points” button is used (Figure 3.2). By pressing this button users are asked to define four image points in order to specify the planar area and perform necessary cropping and rectification operations. After a planar facet in the scene is identified, the system estimates the parameters of the plane (angle between the plane normal and the camera) and the polygon is then simply projected from the image to a new rectangular plane. Rectified image results are displayed in the secondary image area (Figure 3.3) and users can decide whether to use or deny the rectified result as a texture candidate.



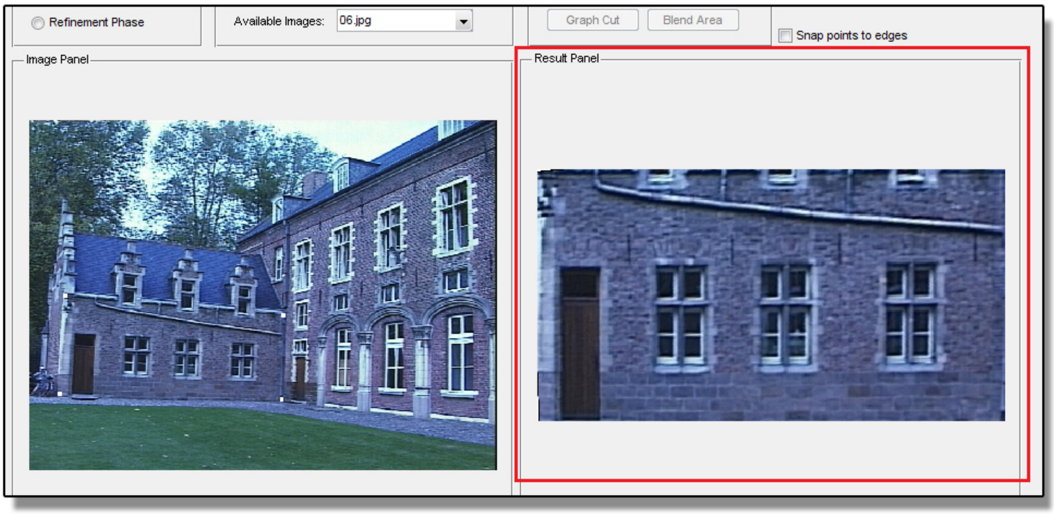
Model Planars: First Wall

Available Images: 01.jpg

□ Snap points to edges

Figure 3.2. User interface elements used in image rectification.

In the selection mode, a Harris corner detector [67] is implemented as a point selection assistant in order to simplify the selection of corners by snapping the selection of the user to the nearest corner found (Figure 3.2).



○ Refinement Phase

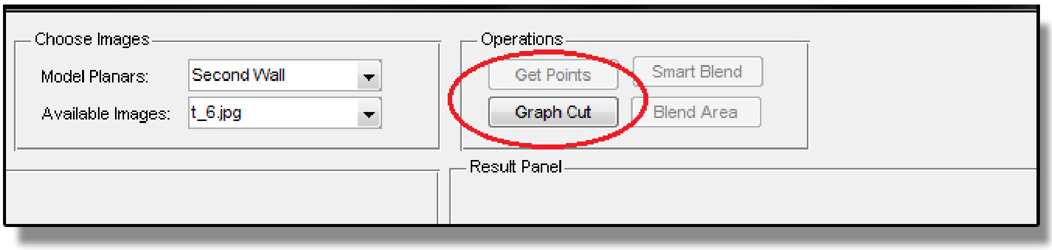
Available Images: 06.jpg

□ Snap points to edges

Figure 3.3. Image area for displaying image editing results.

3.2.2. Refinement Mode

Having selected the candidate images for all or part of the planar surfaces, one can move on with the refinement mode. In the refinement mode the user interface

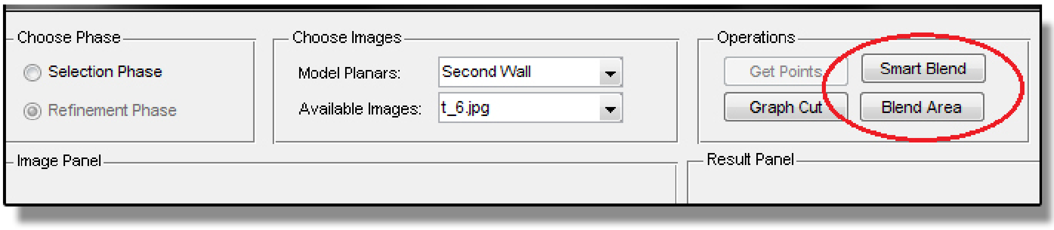


Model Planars: Second Wall

Available Images: t_6.jpg

Figure 3.4. Menu situation of the user interface during refinement phase.

provides the tools that enable the composition the resulting texture map. Firstly, the plane whose texture will be created is selected. Then Graph Cut procedure is applied using the candidate images selected and rectified before by pressing the button "Graph Cut" (Figure 3.4). The resulting labeling and corresponding image are displayed in first and secondary image areas in order. At this step, three options are provided to the user: 1) User can accept the resulting composite as the final texture of the target plane, 2) User can specify undesired areas like occlusions so that the system can provide different composites by excluding the specified areas, 3) If the user is not satisfied with the results of Graph Cut algorithm he/she will apply either local blending by selecting seam area or smart blending by leaving the blending to an automatic process (Figure 3.5).



○ Selection Phase

● Refinement Phase

Model Planars: Second Wall

Available Images: t_6.jpg

Figure 3.5. Menu situation of the user interface after Graph Cut.

Another handy tool implemented to reduce the texture mapping time is the substitution tool. This property lets the user to use a predefined texture for another plane in the model. Thus, using the combo box (Figure 3.6) including the predefined textures, users do not lose time by texturing the similar surfaces again and again.

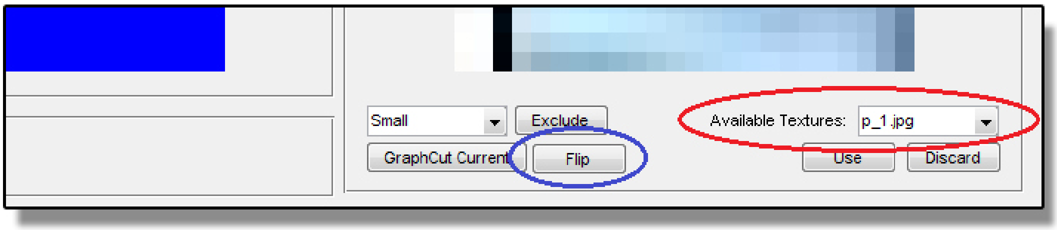


Figure 3.6. Red circle marks the predefined textures that can be used again. Blue circle marks the flip property.

In addition, a “Flip” feature (Figure 3.6) is added to provide the opportunity to flip final images while texturing symmetric planes.

3.2.3. 3D Model Viewer

The 3D viewer is separated from the 2D user interface. The user can view the current status of the 3D model at any phase of the texture creation cycle by using a 3D viewer which includes only the completed planar textures.

4. EXPERIMENTS AND RESULTS

4.1. Datasets and Framework

All the algorithms described in this thesis, except the 3D model viewer (Section 3.2.3), are implemented in MATLAB 7. Some useful code libraries are imported into the developed system eighter by partly or totally while the rest of the problems are solved in the boundaries of the MATLAB framework. Graph cut optimization is one of the concepts that we employed a library [73] to handle. In fact, we used a MATLAB wrapper [74] which is first implemented in [75].

We have also applied a modified version of the Poisson solver implemented in [78] and used in gradient domain fusion algorithms applied for texture refinement like seamless cloning.

All algorithms and techniques proposed in this thesis are combined and presented in a user friendly interface which is also developed in the MATLAB framework.

Furthermore a 3D model viewer (Section 3.2.3) is implemented in the Adobe Flash platform by defining an XML based model structure composed of planes. All 3D models used in this thesis are created using this structure. Required 3D engine is taken from Papervision3D [79] which is an open source 3D engine for the Flash platform.

For all examples used in this chapter, photographs are acquired with a standard consumer level digital camera with no additional setup or professional assistance. As stated in previous sections (Sections 2.1 and 2.2) we did not make use of any digitally available feature like camera parameters.

4.2. Experiments

Since the aim of this thesis is to texture map 3D building models using unordered photo collections, we exemplify the concept with different types of buildings and state the results in this part. Using our system, we have textured a variety of scenes ranging from small houses to complex architecture. Tables 4.1, 4.4, 4.8 sample the photos used to texture map each structure while tables 4.2, 4.3, 4.5, 4.9 report the number of photos used and the time spent by the user while texturing the models in our system. For all the datasets except for the first example (Leuven Castle) which contains detailed microstructures, the user was able to texture a coarse model of the whole scene very quickly. Thus, some results (Table 4.3) do not require much interaction by the user; textures are selected automatically by only graph cut process. Other results, on the other hand, require some modifications by user driven information (Tables 4.5, 4.9).

This section includes detailed results of experiments applied on three different building models. The first image set is obtained from [80] and includes 28 photographs with almost no occluders. The images belong to a historical building called *Leuven Castle* which has a quite complex model with lots of detailed structures.

The second image sequence is taken from an image database created for multiview stereo 3D reconstruction [81]. We make use of the image set *Castle R20* (original name of the building is not given) which contains 20 photographs taken from the center of the rectangular building structure.

We have produced the last image sequence by taking ground-based images of *Albert Long Hall*, another historical building located in Boğaziçi University, Istanbul. Neighbouring buildings and different types of occluders appearing in the image set, make the building a good candidate for experimentation.

In the Leuven Castle sequence (Table 4.1), two roof planes are visible but since they include microstructures it is hard to extract accurate texture especially for the higher roof. On the other hand, the Leuven Castle sequence includes many replication

Table 4.1. Some of the photographs used to texture map the Leuven Castle.

of planes because of the repeating windows and microstructures. As a result, although the model is quite complex and the number of total planes is high, by using texture duplicates in more than one plane, texture mapping time decreases significantly. With respect to other examples stated in this section, this image set does not contain foreground occluders except for the bicycles located next to one of the walls. This occlusion is left as a part of the wall texture since it is impossible to remove such nearly adjacent occlusions unless manual and professional user interaction is provided which is out of the target of our system.

Here, we have created two different models of the same building with different level of details. As stated in Tables 4.2 and 4.3 it takes almost 5 min to texture map the simple model with 15 planes. All detailed microstructures like rooftop windows are removed in this simple model. However, texturing the complex model takes nearly 15 min with 46 planes. This result seems to be quite acceptable compared to the results of [12] with the same building by taking into consideration that their results also include modeling times.

Castle R20 image sequence includes photographs taken in the middle of a rectangular shaped building. Table 4.4 shows some of the photographs included in this dataset. Obviously the tractor seen in almost every photograph becomes an occluder for most of the planes in 3D model. Since the buildings in this set are quite high and all images in the set are close range, we choose not to model rooftops since we could not get an acceptable quality.

Necessary statistics about this model are given in Table 4.5. In addition, as seen in the model views and specifically in Table 4.6, occlusions caused by the tractor are removed in almost all textures. In Table 4.6 first row includes three different views of the same wall. Second row specifies the calculated angles between viewing angles and the plane normals while the third row contains transformed images after perspective distortion is removed. Finally, the last row represents initial labeling before the graph cut process, final labeling and final texture composite respectively.

Table 4.2. 3D detailed model of *Leuven Castle* with necessary statistics.

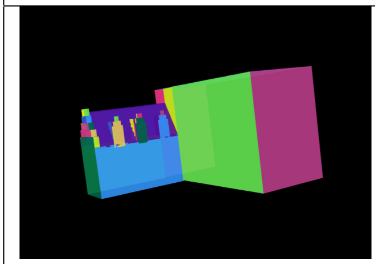
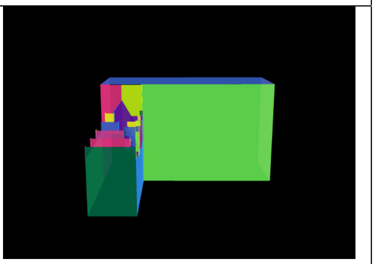
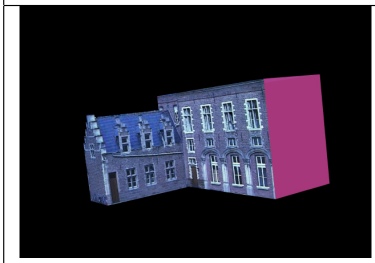
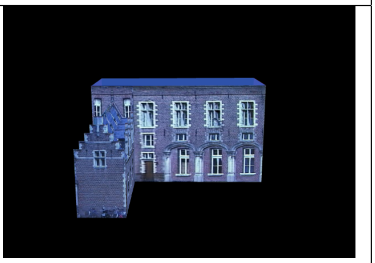
Model: Leuven Castle (Detailed)	
Number of photographs:	28
Number of planes in model:	46
Number of visible planes:	41
Texture mapping time:	18 min
Untextured model views	
	
Textured model views	
	

Table 4.3. 3D simple model of *Leuven Castle* with necessary statistics.

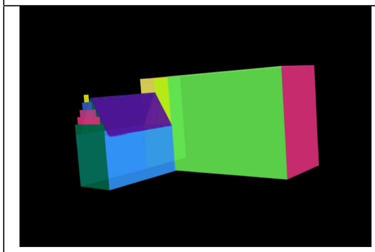
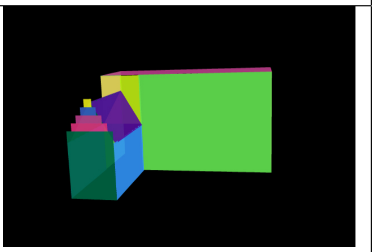
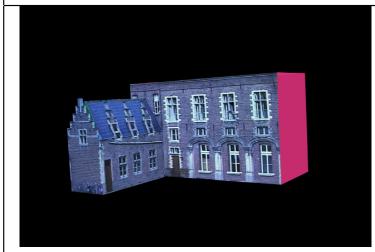
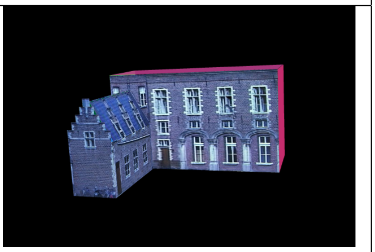
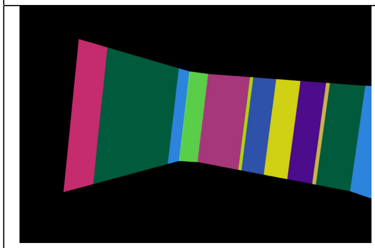
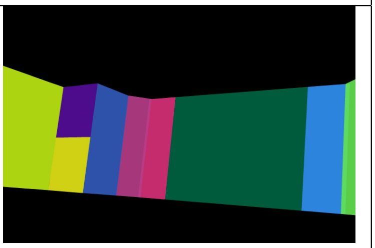
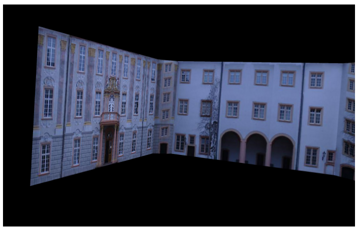
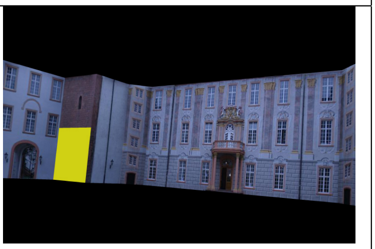
Model: Leuven Castle (Simple)	
Number of photographs:	28
Number of planes in model:	15
Number of visible planes:	10
Texture mapping time:	5 min
Untextured model views	
	
Textured model views	
	

Table 4.4. Some of the photographs used to texture map the Castle R20.

Table 4.5. 3D model of *Castle R20* with necessary statistics.

Model: Castle R20	
Number of photographs:	20
Number of planes in model:	17
Number of visible planes:	16
Texture mapping time:	11 min
Untextured model views	
	
Textured model views	
	

There is also a green box occluding some part of the walls in the right side of the building. This occluder is also removed by using duplicate textures. Table 4.7 shows the stages of this operation by giving available dataset images in the first row in which the target plane is focused by red outlines. Here, green outlines represents the problematic occlusions with different kinds of occluders (tractor, ivy and green waste bin). In the second row of the same table, we exemplify the usage of the same texture for more than one plane to compose a complete nonoccluded texture.

The last example is taken from Albert Long Hall which is located in Boğaziçi University, Istanbul. Table 4.8 includes some of the photographs used in texture mapping of this building. In total, 13 photographs, acquired with a standard digital camera, have been used in the process. Because of the differing sunlight directions, intensity values change significantly in this image set. There are also lots of occluders like the

Table 4.6. Occlusion removal during texture mapping of Castle R20.

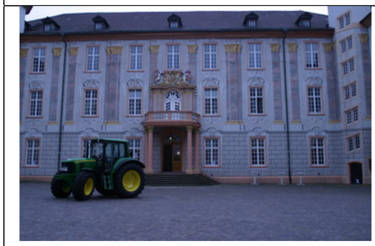
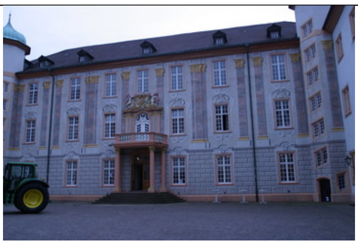
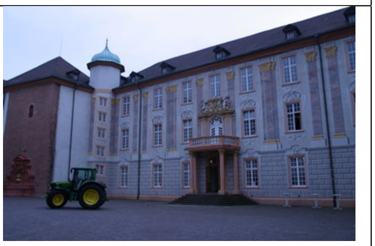
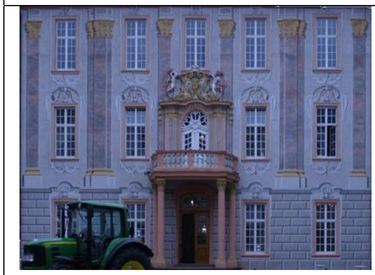
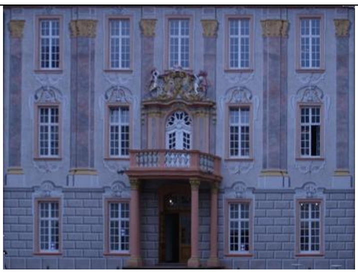
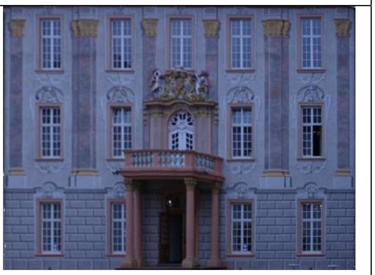
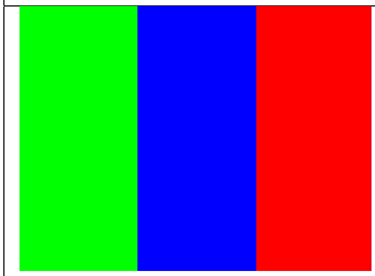
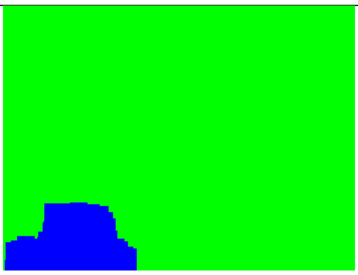
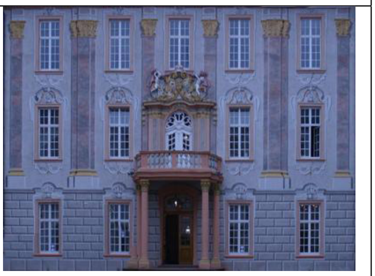
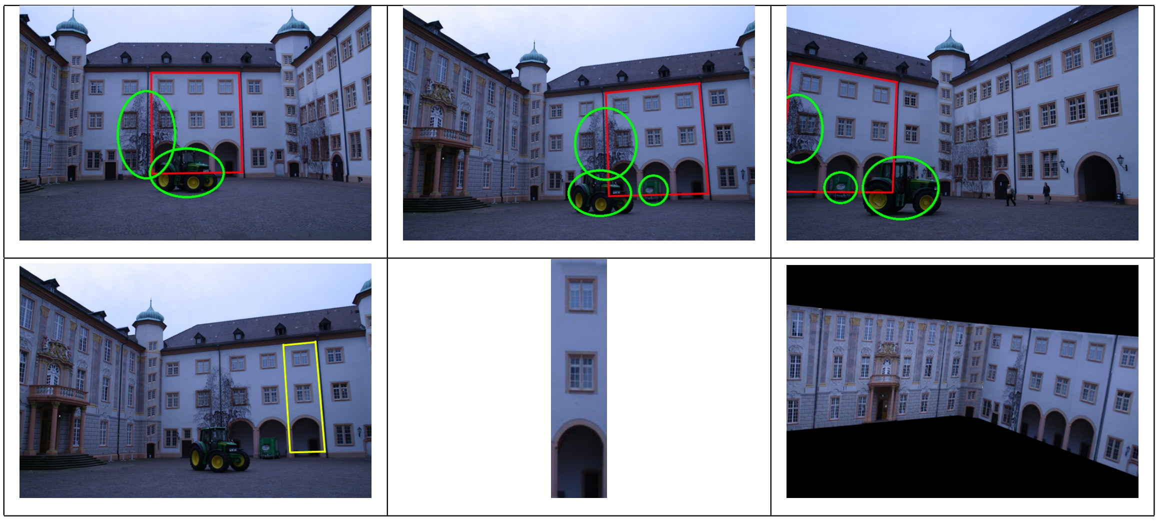
Unordered image samples		
		
Angles estimated		
2	57	76
Rectified texture candidates		
		
Initial labeling, final labeling and final texture composite		
		

Table 4.7. Using duplicate surfaces during texture mapping of Castle R20.

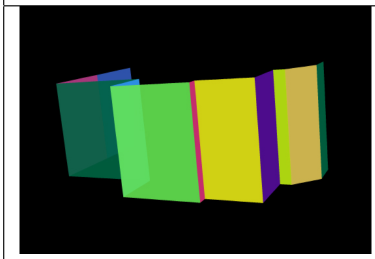
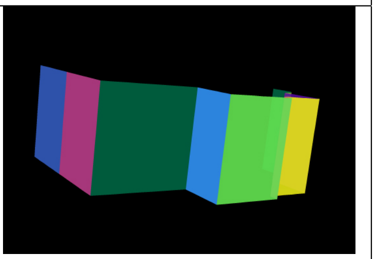
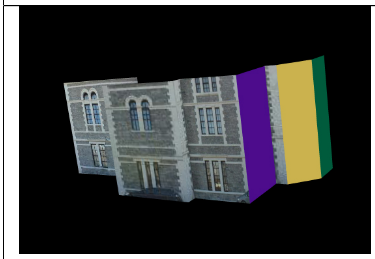
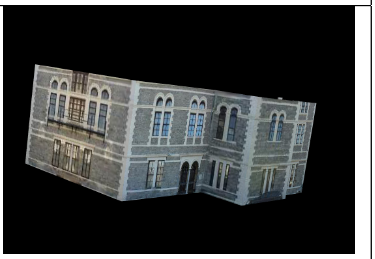


bushes and the cars in front of the entrance which have been removed during graph cut process by using visible textures for the occluded regions. On the other hand, there are also adjacent occluders like the fire escape on the left side and the trees on the right side. We handle the occlusion caused by fire escape structure with the help of flip and duplicate tools we provided in the user interface. Symmetrical structure of this side let us create a clear texture by using the non-occluded part of the symmetry. However, the situation is not valid for the right hand side of the building since this side does not include such a symmetric structure and has adjacent occluders composed of large trees. So we decided to leave this side as it is with the occluders in the final model represented in Table 4.9.

As seen, the amount of user interaction after graph cut process directly increases total texture mapping time. However, user driven information mostly provides more realistic results. Examples stated here are described as simple or complex according to the number of microstructures on the models and occlusions in the datasets. Moreover, although a model includes many microstructures, most of these structures may repeat throughout the building. Thus, for any model, the shape and the properties like symmetry highly effect texture mapping times.

Table 4.8. Some of the photographs used to texture map the Albert Long Hall.

Table 4.9. 3D model of *Albert Long Hall* with necessary statistics.

Model: Albert Long Hall	
Number of photographs:	13
Number of planes in model:	11
Number of visible planes:	8
Texture mapping time:	6 min
Untextured model views	
	
Textured model views	
	

5. CONCLUSIONS

A semi-automatic texture mapping system has been proposed in this thesis. This system makes use of unordered photos of a real world object in order to texture map the 3D model of that object. Projective Transformation, Graph Cut Optimization, Poisson Blending and some helpful user interface tools have been used to cope with different problems that create the texture mapping cycle. Defining these problems as different phases of the main texture mapping problem we divide the overall system into three main steps which can be ordered as image rectification, creating the optimal texture composite and texture refinement. Based on the experiments and results reached in this thesis, conclusions for each step are summed up in the following categories:

Image Rectification: Since we do not make use of any predefined parameters like camera position we built up a rectification system by estimating the perspective distortions existing in the images according to the plane outlines drawn by the users. Although it is a weak approach whose accuracy depends on the accuracy of the outlines, we implement a corner detector to help the user through the drawing process. Moreover, we noticed that small shifts/errors on the estimated angles do not cause big changes in the resulting texture composites. Thus, we decided to move on with the estimated plane angles instead of making use of predefined camera parameters which inserts a qualified user into the loop.

Creating The Optimal Texture Composite: We have seen that by using graph cut optimization we get rid of most of the refinement steps, especially for the cases where we have lots of candidate images. Thus, in most cases graph cut optimization provides us the proper texture composites to go over or employ some refinements. The main criteria focused on this work was the color consistency of the candidates and the preference of frontal views. However, for specific purposes, one can add many other criteria like luminance or contrast. Color consistency criterion helps removing most of the occluders automatically unless the occluder contains very similar colors with the plane it has occluded. Thus, this kind of problematic occluders are handled with user intervention using the exclude tool.

Texture Refinement: The main technique we used in texture refinement phase is gradient domain fusion which is used in Poisson image editing. By applying the idea of seamless cloning in texture mapping we created a tool called smart blend to employ seamless cloning for each fragment of the texture composites provided by graph cut optimization.

There are also assistant tools like the substitute tool that enables the user to use a texture for more than one plane or the flip tool which is also helpful in texturing symmetrical surfaces. There is also an exclude tool defined to remove undesired surfaces (generally occluders) from a texture composite.

All these refinement tools help users to develop a better texture composite if they are not satisfied with the results of graph cut texture composites. The number of refinement tools provided can be increased by including other traditional tools like clone brushing or color averaging. However, although these tools enhance the capabilities of users, they also increases the texture mapping times by increasing the level of user interaction.

5.1. Remarks and Future Directions

In this thesis, we mainly focused on texture mapping of a given 3D model. Obviously, texture mapping is important while creating photorealistic and detailed 3D models. It is also necessary to have specific information about the targeted object like cultural heritage. Therefore texture mapping constitutes an important role in the 3D reconstruction problem and the whole system proposed here can also be considered as a module of a bigger 3D reconstruction system.

Acquiring photorealistic 3D models of real world objects is a growing application area and becoming target of increasing interest. Thus, researchers gradually integrate traditional image processing methods into this area by modifying the algorithms according to the needs of 3D reconstruction. Therefore, considering this thesis, there are still some open questions which can be subjects of further research.

One issue is that using a semi-automatic system, texture synthesis approaches can be applied to texture some parts which are invisible in all the dataset images used. [49] proposes such an approach to texture map invisible areas.

Although graph cut optimization provides seamless textures by including color penalties, preserving global illumination through the entire 3D model is another task that can be helpful to provide better results especially for the photographs taken in different times of the day. This problem also arises another issue which is about the ease of access. After the texture mapping process is completed, all relevant parts of the original images can be packed into one single large texture image to provide easier handling for global modifications.

The proposed approach is presented with a detailed texture mapping application which provides all algorithms as simple tools to let ordinary users with no professional experience to handle the entire texture mapping process. Transforming this application into a 3D editing tool and integrating into a 3D reconstruction system may also be stated as a future work. Thus such a complete system, with provided camera parameters, will increase the quality of results reached in this thesis.

REFERENCES

1. Debevec, P. E., C. J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach", *SIGGRAPH 1996, Computer Graphics Proceedings*, pp. 11–20, 1996.

2. Pollefeys, M., D. Nister, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, C. Salmi, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, and H. Towles, "Detailed Real-Time Urban 3D Reconstruction from Video", *International Journal of Computer Vision*, Vol. 78, pp. 143–167, 2008.

3. Goesele, M., N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-View Stereo for Community Photo Collections", *IEEE International Conference on Computer Vision*, Vol. 0, pp. 1–8, 2007.

4. Snavely, N., S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections In 3D", *ACM Transactions on Graphics (SIGGRAPH)*, pp. 835–846, 2006.

5. Dick, A. R., P. H. S. Torr, and R. Cipolla, "Modelling and Interpretation of Architecture from Several Images", *International Journal of Computer Vision*, Vol. 60, No. 2, pp. 111–134, 2004.

6. Werner, T. and A. Zisserman, "New Techniques for Automated Architecture Reconstruction from Photographs", *Proceedings of the 7th European Conference on Computer Vision*, Vol. 2, pp. 541–555, 2002.

7. Cipolla, R. and D. Robertson, "3D Models of Architectural Scenes from Uncalibrated Images and Vanishing Points", *International Conference on Image Analysis and Processing*, Vol. 0, p. 824, 1999.

8. Cipolla, R., D. Robertson, and E. Boyer, "PhotoBuilder - 3D Models of Architec-

tural Scenes from Uncalibrated Images", *International Conference on Multimedia Computing and Systems*, Vol. 1, p. 9025, 1999.

9. Oh, B. M., M. Chen, J. Dorsey, and F. Durand, "Image-Based Modeling and Photo Editing", *Proceedings of ACM SIGGRAPH'01*, pp. 433–442, 2001.

10. El-Hakim, S., E. Whiting, and L. Gonzo, "3d Modeling with Reusable and Integrated Building Blocks", *The 7th Conference on Optical 3-D Measurement Techniques*, 2005.

11. Hengel, A., A. Dick, T. Thormählen, B. Ward, and P. Torr, "VideoTrace: Rapid interactive scene modelling from video", *ACM Transactions on Graphics (TOG), ACM SIGGRAPH*, Vol. 26, 2007.

12. Sinha, S. N., D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3D Architectural Modeling From Unordered Photo Collections", *ACM Transactions on Graphics (SIGGRAPH)*, Vol. 27, pp. 1–10, 2008.

13. Criminisi, A., I. Reid, and A. Zisserman, "Single View Metrology", *International Journal of Computer Vision*, Vol. V40, pp. 123–148, 2000.

14. Wilczkowiak, M., P. Sturm, and E. Boyer, "Using Geometric Constraints through Parallelepipeds for Calibration and 3D Modeling", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, pp. 194–207, 2005.

15. Frueh, C., R.Sammon, and A.Zakhor, "Automated Texture Mapping of 3D City Models With Oblique Aerial Imagery", *The 2nd International Symposium on 3D Data Processing Visualization and Transmission*, pp. 396–403, 2004.

16. Akbarzadeh, A., J. m. Frahm, P. Mordohai, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistér, and M. Pollefeys, "Towards Urban 3D Reconstruction From Video", *In Proceedings of the Third International Symposium on*

3D Data Processing, Visualization and Transmission (3DPVT), pp. 1–8, 2006.

17. Stamos, I. and P. Allen, “Geometry and Texture Recovery of Scenes of Large Scale”, *Computer Vision and Image Understanding*, Vol. 88, pp. 94–118, 2002.

18. Gibson, S., R. J. Hubbold, J. Cook, and T. L. J. Howard, “Interactive Reconstruction of Virtual Environments from Video Sequences”, *Computers and Graphics*, Vol. 27, pp. 293–301, 2003.

19. Tsai, F., C.-H. Chen, J.-K. Liu, and K.-H. Hsiao, *Texture Generation and Mapping Using Video Sequences for 3D Building Models*, chapter Lecture Notes in Geoinformation and Cartography: Innovations in 3D Geo Information Systems, pp. 429–438, Springer Berlin Heidelberg, 2006.

20. Wu, J., G. Zhou, F. Ding, and Z. Liu, “Automatic Retrieval of Optimal Texture from Aerial Video for Photo-realistic 3D Visualization of Street Environment”, *Image and Graphics, International Conference on*, pp. 943–947, 2007.

21. Lee, S. C., S. K. Jung, and R. Nevatia, “Automatic Integration of Facade Textures into 3D Building Models with a Projective Geometry Based Line Clustering”, *Computer Graphics Forum*, Vol. 21, pp. 259–273, 2002.

22. Frueh, C. and A. Zakhor, “Constructing 3D City Models by Merging Ground-Based and Airborne Views”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, p. 562, 2003.

23. Coorg, S. and S. Teller, “Spherical Mosaics with Quaternions and Dense Correlation”, *International Journal of Computer Vision*, Vol. 37, pp. 259–273, 2000.

24. Efros, A. A. and W. T. Freeman, “Image Quilting For Texture Synthesis and Transfer”, *SIGGRAPH’01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 341–346, ACM, New York, NY, USA, 2001.

25. Zhu, Z., E. M. Riseman, and A. R. Hanson, “Parallel-Perspective Stereo Mosaics”,

In IEEE International Conference on Computer Vision, pp. 345–352, 2001.

26. Zheng, J. Y. and M. Shi, “Mapping Cityscapes to Cyber Space”, *Proceedings of the 2003 International Conference on Cyberworlds*, p. 166, IEEE Computer Society, Washington, DC, USA, 2003.

27. Spann, J. and K. Kaufman, “Photogrammetry Using 3D Graphics and Projective Textures”, *IAPAS, Amsterdam*, Vol. 33, 2000.

28. Kumar, R., H. Sawhney, Y. Guo, S. Hsu, and S. Samarasekera, “3D Manipulation of Motion Imagery”, *International Conference on Image Processing (ICIP)*, Vol. 1, pp. 17–20, 2000.

29. Rother, C., “A New Approach for Vanishing Point Detection in Architectural Environments”, *In Proceedings of 11th British Machine Vision Conference*, pp. 382–391, 2000.

30. Pulli, K., M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle, “View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data”, *In Eurographics Rendering Workshop*, pp. 23–34, 1997.

31. Debevec, P., Y. Yu, and G. Borshukov, “Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping”, *In 9th Eurographics Rendering Workshop*, pp. 14–26, Springer, 1998.

32. Lensch, H. A., W.Heidrich, and H. Seidel, “Automated Texture Registration and Stitching for Real World Models”, *Eighth Pacific Conference on Computer Graphics and Applications*, pp. 317–452, 2000.

33. Porquet, D., J.-M. Dischler, and D. Ghazanfarpour, “Real-Time High-Quality View-Dependent Texture Mapping Using Per-Pixel Visibility”, *GRAPHITE ’05: Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pp. 213–220, ACM, 2005.

34. Rocchini, C., P. Cignoni, C. Montani, and R. Scopigno, "Multiple Textures Stitching and Blending on 3D Objects", *Eurographics Rendering Workshop*, pp. 119–130, 1999.

35. Baumberg, A., "Blending Images for Texturing 3D Models", *Proceedings of the British Machine Vision Conference*, pp. 404–413, 2002.

36. Wang, X., S. Totaro, F. Taillandier, A. R. Hanson, and S. Teller, "Recovering Facade Texture and Microstructure", *2nd International Workshop on Texture Analysis and Synthesis at ECCV*, pp. 145–149, 2002.

37. Chon, J., T. Fuse, and E. Shimizu, "Urban Visualization Through Video Mosaics Based On 3-D Multibaselines", *International Archive of Photogrammetry and Remote Sensing*, Vol. XXXV-B3, pp. 727–731, 2004.

38. Zhang, Y., Z. Zhang, J. Zhang, and J. Wu, "3D Building Modelling with Digital Map, Lidar Data and Video Image Sequences", *The Photogrammetric Record*, Vol. 20(111), pp. 285–302, 2005.

39. Besag, J., "On the Statistical Analysis of Dirty Pictures", *Journal of the Royal Statistical Society*, Vol. B-48, pp. 259–302, 1986.

40. Barker, S. and P. Rayner, "Unsupervised Image Segmentation", *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 5, pp. 2757–2760, 1998.

41. Horn, B. K. P. and B. G. Schunck, "Determining Optical Flow", *Artificial Intelligence*, Vol. 17, pp. 185–203, 1981.

42. Chou, P. B. and C. M. Brown, "The Theory and Practice of Bayesian Image Labeling", *International Journal of Computer Vision*, Vol. 4, No. 3, pp. 185–210, 1990.

43. Szeliski, R., "Bayesian Modeling of Uncertainty in Low-Level Vision", *International*

Journal of Computer Vision, Vol. 5, No. 3, pp. 271–302, December 1990.

44. Amini, A. A., T. E. Weymouth, and R. C. Jain, "Using Dynamic Programming for Solving Variational Problems in Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 9, pp. 855–867, 1990.

45. Boykov, Y., O. Veksler, and R. Zabih, "Fast Approximate Energy Minimization via Graph Cuts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, pp. 1222–1238, 2001.

46. Roy, S. and I. J. Cox, "A Maximum-Flow Formulation of the N-Camera Stereo Correspondence Problem", *Proceedings of the Sixth International Conference on Computer Vision*, p. 492, IEEE Computer Society, Washington, DC, USA, 1998.

47. Ishikawa, H. and D. Geiger, "Occlusions, Discontinuities, and Epipolar Lines in Stereo", *In European Conference on Computer Vision*, pp. 232–248, 1998.

48. Boykov, Y., O. Veksler, and R. Zabih, "Markov Random Fields With Efficient Approximations", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–655, 1998.

49. Kwatra, V., A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts", *ACM Transactions on Graphics*, Vol. 22, pp. 277–286, 2003.

50. Agarwala, A., M. Dontcheva, M. Agrawala, S. M. Drucker, A. Colburn, B. Curless, D. Salesin, and M. F. Cohen, "Interactive Digital Photomontage.", *ACM Transactions on Graphics*, Vol. 23, pp. 294–302, 2004.

51. Ogden, J. M., E. H. Adelson, J. R. Bergen, and P. Burt, "Pyramid-Based Computer Graphics", *RCA Engineer*, Vol. 30(5), pp. 4–15, 1985.

52. Burt, P. J. and R. J. Kolczynski, "Enhanced Image Capture Through Fusion", *IEEE International Conference on Computer Vision (ICCV 1993)*, pp. 173–182,

1993.

53. Fattal, R., D. Lischinski, and M. Werman, "Gradient Domain High Dynamic Range Compression", *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 249–256, 2002.

54. Pérez, P., M. Gangnet, and A. Blake, "Poisson Image Editing", *ACM Transactions on Graphics (SIGGRAPH 2003)*, Vol. 22, pp. 313–318, 2003.

55. Elder, J. and R. M. Goldberg, "Image Editing in the Contour Domain", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, pp. 291–296, 2001.

56. Lewis, J., "Lifting Detail from Darkness", *SIGGRAPH 2001 Tech Sketch*, 2001.

57. *ADOBE ©2002 Photoshop ©7.0 User Guide*.

58. Burt, P. J. and E. H. Adelson, "A multiresolution spline with application to image mosaics", *ACM Transactions on Graphics*, Vol. 2, pp. 217–236, 1983.

59. Adelson, E. H., C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid Methods in Image Processing", *RCA Engineer*, Vol. 29, No. 6, pp. 33–41, 1984.

60. Levin, A., A. Zomet, S. Peleg, and Y. Weiss, "Seamless Image Stitching in the Gradient Domain", *Eighth European Conference on Computer Vision*, pp. 377–389, 2004.

61. Bertalmio, M., G. Sapiro, V. Caselles, and C. Ballester, "Image Inpainting", *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 417–424, 2000.

62. Barrett, W. A. and A. S. Cheney, "Object-Based Image Editing", *ACM Transactions on Graphics 21*, Vol. 3, pp. 777–784, 2002.

63. Bornard, R., E. Lecan, L. Laborelli, and J.-H. Chenot, “Missing Data Correction in Still Images and Image Sequences”, *MULTIMEDIA ’02: Proceedings of the Tenth ACM International Conference on Multimedia*, pp. 355–361, ACM, New York, NY, USA, 2002.

64. Du, Y., J. Cihlar, J. Beaubien, and R. Latifovic, “Radiometric Normalization, Compositing, and Quality Control For Satellite High Resolution Image Mosaics Over Large Areas”, *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 39, pp. 623–634, 2001.

65. Xiao, J., T. Fang, P. Tan, P. Zhao, E. Ofek, and L. Quan, “Image-Based Façade Modeling”, *ACM Transactions on Graphics*, Vol. 27, No. 5, pp. 161:1–161:10, December 2008.

66. Criminisi, A., P. Pérez, and K. Toyama, “Object Removal by Exemplar-Based Inpainting”, *In Proceedings of IEEE Computer Vision and Pattern Recognition*, Vol. 2, pp. 721–728, 2003.

67. Harris, C. and M. Stephens, “A Combined Corner and Edge Detector”, *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.

68. Liebowitz, D. and A. Zisserman, “Metric Rectification for Perspective Images of Planes”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 482–488, 1998.

69. Hartley, R. and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second edition, 2004.

70. Lempitsky, V. S. and D. V. Ivanov, “Seamless Mosaicing of Image-Based Texture Maps”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

71. Ford, L. R. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press,

Princeton, N.J.,, 1962.

72. Zeleznik, R. C., K. P. Herndon, and J. F. Hughes, "SKETCH: An Interface for Sketching 3D Scenes", *Proceedings of SIGGRAPH 96*, pp. 163–170, 1996.

73. Veksler, O., "Software for Energy Minimization with Graph Cuts, Version 1.3", 1999, `http://www.csd.uwo.ca/faculty/olga/code.html`.

74. Bagon, S., "Matlab Wrapper for Graph Cut Optimization Software", December 2006, `http://www.wisdom.weizmann.ac.il/∼bagon`.

75. Veksler, O., *Efficient Graph-Based Energy Minimization Methods In Computer Vision*, Ph.D. thesis, Cornell University, July 1999.

76. Kolmogorov, V. and R. Zabih, "What Energy Functions Can Be Minimized via Graph Cuts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, pp. 147–159, 2004.

77. Boykov, Y. and V. Kolmogorov, "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 26, pp. 1124–1137, 2004.

78. Zhou, H. and J. Sun, "Code Library for Poisson Image Editing Problems", 2004, `http://www.howardzzh.com/research/poissonImageEditing`.

79. Ulloa, C., J. Grden, R. Hauwert, T. Knip, and A. Zupko, "Papervision3D - Public Beta 2.0 ©2006-2008", `http://code.google.com/p/papervision3d/`.

80. Pollefeys, M., "Image Sequence of Leuven Castle Located in Leuven, Belgium", `http://www.cs.unc.edu/∼marc/`.

81. Strecha, C., W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen, "Online Multiview Stereo Datasets", `http://cvlab.epfl.ch/∼strecha/multiview/`.

82. Strecha, C., W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen, “On Benchmarking Camera Calibration and Multi-View Stereo For High Resolution Imagery”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 0, pp. 1–8, 2008.