MIT

The City Scanning Project : Validation and Parallel Algorithms

Author : Olivier Koch
Advisor : Pr. Seth Teller

MIT Computer Graphics Group

http://graphics.lcs.mit.edu
olivier.koch@ensta.org

Abstract

Extract from the City Scanning Project Web Site (available at [1]):
"The "computer vision" or "machine vision" problem is a long-standing, difficult problem. Fundamentally, it is: how can a computer algorithm, alone or in concert with a human, produce a useful computational representation of a scene, given one or more images of that scene? Until now, whether automated or human-assisted, no acquisition system has been demonstrated to scale to spatially extended, complex environments, under uncontrolled lighting conditions (i.e., outdoors) and in the presence of severe clutter and occlusion. Given images of an urban environment, we wish to produce a textured geometric CAD model of the structures in that environment. This model should include geometric representations of each structure and structural feature present, and radiometric information about the components of each such entity (e.g., a texture map or bi-directional reflectance distribution function for each surface). One might think of each image as a 2D "observation" of the world; we wish to produce a single, textured 3D CAD model which is consistent with the totality of our observations. Moreover, we wish to achieve a model accurate to a few centimeters, over thousands of square meters of ground area. Several processes are very time-consuming. Reconstruction of the buildings can take several hours on a MIPS 1000 processor. The purpose of our work therefore is to map the algorithmic components of the City Scanning Project onto a cluster of 32 Linux machines."

The first section gives a short overview of the City Scanning Project. We then present our work on code validation and parallel algorithms. In sections 4 and 5, we describe our work on camera pose refinement, as well as several ideas which have been proposed during our stay. Last section contains a short description of the MIT Computer Graphics Group.

Comment une machine peut-elle, seule ou aidée d'un opérateur humain, concevoir la représentation d'une scène à partir de photographies ? De cette question est né le City Scanning Project, dirigé par le professeur Seth Teller, Computer Graphics Group, Massachusetts Institute of Technology. À partir de photographies calibrées prises dans un environnement urbain, le projet a pour but de reconstruire un modèle en trois dimensions de cet environnement. Le modèle comprend une représentation géometrique ainsi que des informations sur la texture et l'éclairement des objets. En considérant chaque image comme une "observation" de l'environnement, le modèle final doit être conforme à l'ensemble de ces observations. La précision du modèle est de l'ordre de quelques centimètres pour une surface d'environ cent mètres carrés.

Le projet est constitué de nombreux algorithmes dont certains sont très coûteux en temps de calcul. La reconstruction géometrique, par exemple, peut durer sept heures sur une machine Silicon Graphics 02 monoprocesseur. Le but principal de mon projet est de paralléliser ces calculs sur les différents clusters du groupe.

Dans un premier temps, nous décrivons le projet de manière globale, pour présenter dans un deuxième temps nos travaux sur la validation du code et le calcul parallèle. Ce document évoque ensuite les travaux annexes qui ont été effectués ainsi que des idées proposées en cours d'ouvrage. Le lecteur curieux trouvera en dernière partie des informations sur le MIT Computer Graphics Group.

Contents

1 Introduction to the City Scanning Project

1.1 Theory : why is it likely to work ?

In this section, we present some very basic geometric ideas. The question is the following : what makes us believe that a set of cameras is sufficient to give a complete description of a 3D environment ? After all, a camera has a 2D representation of the world. How is it possible that a combination of these cameras can create a relevant 3D world ? In fact, a finite set of camera will never be able to represent any environment. For many reasons, the situation can be ambiguous, so that the computer system cannot give a complete solution to the problem. For example, if there are too many trees, some part of the buildings might be occluded to the cameras, and then never be detected. However, this is not exactly the point of the project. The aim of the system is to create a 3D environment which is relevant with all the observations. The question is not : is this world true or not ? The question is : is it relevant or not ? Now, if we consider that we have enough observations, this world will be unique, and the relevant world will correspond to te real world, which is not obvious in fact. Thus, given the assumption that the system is intelligent enough to put the camera in the right positions, and a sufficient number of times, we can consider that our system will be able to represent the surrounding urban environment. Let us now consider a camera. It has a 2D representation of the world, which means that the information provided by this camera is not sufficient to reconstruct the 3-dimension scene. However, if we use the right projection, and get rid of the problem of distorsion, a line in the 3D world will be represented by a line on the camera plane. Thus, the camera will be able to detect the lines in our world. At that point, we need a geometrical theorem, which is quite simple in fact ([8]):

Theorem 1.1 *Given the position of the camera in the 3D world, the direction of a 3D-horizontal edge is completely determined by the image-space 2D line of its projection.*

Figure 1 illustrates this idea. Let us give a short proof of the theorem. Let us put C the center of the camera, and B and E the endpoints of the edge on the image plane. C, B and E define a unique plane in the 3D world. Let us call D the intersection of this plane with any horizontal plane. Than D is the direction of the 3D-horizontal edge. Indeed, the 3D edge belongs to plane (B,C,E) and is parallel to any horizontal plane since it is horizontal.

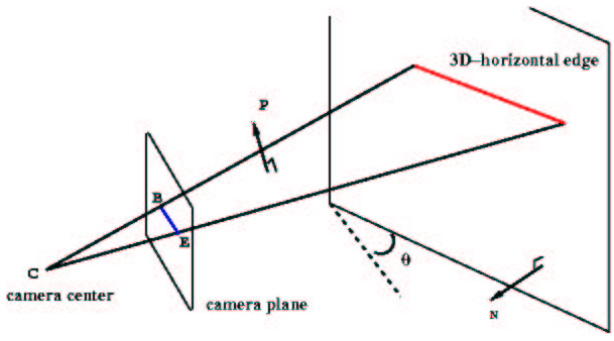

Figure 1: Deducing tile azimuth from a horizontal line segment

Why is this theorem so interesting for the City Scanning Project ? Let us consider a set of cameras taking pictures of the environment and a 3D horizontal edge. Let us assume that this edge

is observed by at least two cameras. According to the last theorem, each camera knows the direction of the edge. If the cameras are "well positioned" in respect with the 3D edge, the edge belongs to the intersection of two planes, as shown on figure 2. Thus, using a space sweep technique, the position of the edge is completely determined. Using this method for all the edges, it is possible to determine all the positions of the edges in the 3D world, and then to reconstruct all the vertical facades, which is an important step in the project. Indeed, vertical facades occur frequently in urban environments.

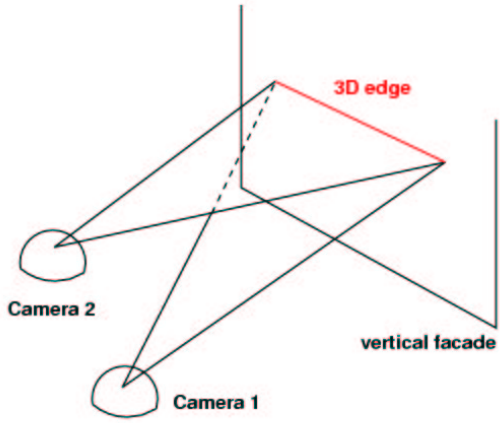

Figure 2: Deducing edge position from multiple observation

As we said, this is a "perfect" situation. In the real life, many aspects make the problem more difficult, and two cameras are unsufficient to detect an edge. The experience tends to prove that, in the case of the City Scanning Project, at least six cameras are necessary. This is mostly due to the fact that the system is fully automated, which means that there is no human operator to help the computer go through ambiguous situations.

1.2 Georeferenced imagery

To observe the urban environment from many angles, the City Scanning Project uses Argus. The device, developed by a team at the MIT Computer Graphics Group, combines navigation sensors, a high-resolution digital camera, and various algorithms to capture georeferenced imagery of urban environments ([3]).

1.2.1 Camera

Figure 3: The eye of Argus

The pose-camera is a wheelchair-sized mobile platform with a high-resolution color digital camera mounted on a pan-tilt head, which is itself mounted on a vertically telescoping stalk. The platform also includes instrumentation to maintain estimates of global positioning (GPS), heading information (IMU), and dead-reckoning (optical encoders on the wheels), all in absolute coordinates (for example, Earth coordinates). Finally, an on-board power source and PC provide power and control to all of the devices, and disk and tape drives store digital image and pose (position, heading, time) data. Each picture taken by the camera is annotated with information such as : date, GPS position, and camera intrinsic parameters (focale, aperture,...). The camera is deployed in the urban area. The human operator chooses the positions or "nodes" from which images are to be acquired, at a spacing of roughly every 10 meters, maintaining a standoff of 10-15 meters from each building. The pose-camera acquires a spherical mosaic image centered at the node, and annotates it with the on-board instrumentation's estimates of latitude, longitude, and altitude of the camera, heading information, and the date and time, and stores the resulting "pose-image."

1.2.2 Images

The acquisition of images and structure of the dataset is fully described in [4]. Figure 4 shows the tiling pattern we use (typically, 47 or 71 images). Why 47 or 71 images? Around the equator, and at 20 and 40 degrees North, we take twelve images spaced 30 degrees apart. At 60 degrees North, we take nine images (spaced 40 degrees apart); at 80 degrees North, only two images (180 degrees apart). The total is then $12 + 12 + 12 + 9 + 2 = 47$ images. Sometimes, nodes are taken from high above ground (e.g., from a nearby building roof, or parking garage top level). These nodes have two extra "rings" of twelve images, at 20 and 40 degrees South, for a total of $47 + 12 + 12 = 71$ images. Note that our "tiling" scheme produces overlapping images; this is necessary for the correlation-based alignment stage to produce spherical mosaics (below). Each "tile" of the mosaic is a high-res (1K by 1.5K by 24-bit) digital image as shown on figure 5. Figure 6 shows what the images look like from a fixed optical center as the camera is rotated about it. Figure 6 also shows the raw images (47 images, 1K x 1.5K x 24 bits each) comprising one node acquired at ground level.

The next step is align each of the images to form a seamless spherical mosaic. We do so by associating a quaternion with each image, then performing numerical optimization to best correlate overlapping regions of neighboring images. Several views of one resulting spherical mosaic are represented on figure 7. For ease of processing, we use a "spherical coordinate" representation of these mosaics. We coalesce each mosaic into a rectangular pixel array whose horizontal coordinate

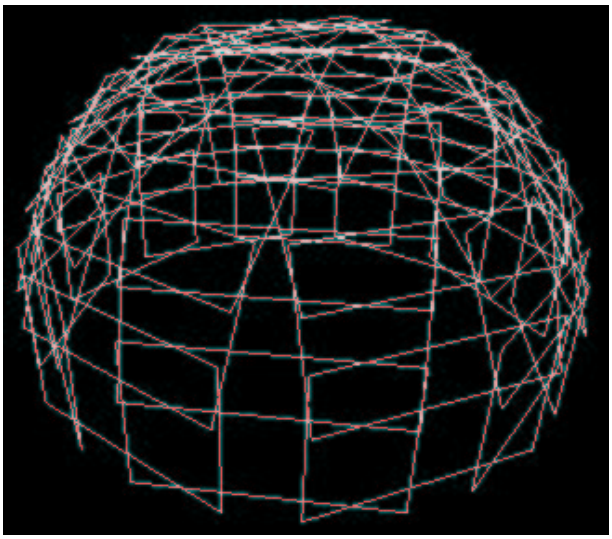

Figure 4: Imagery tiling pattern

Figure 5: High-res digital image

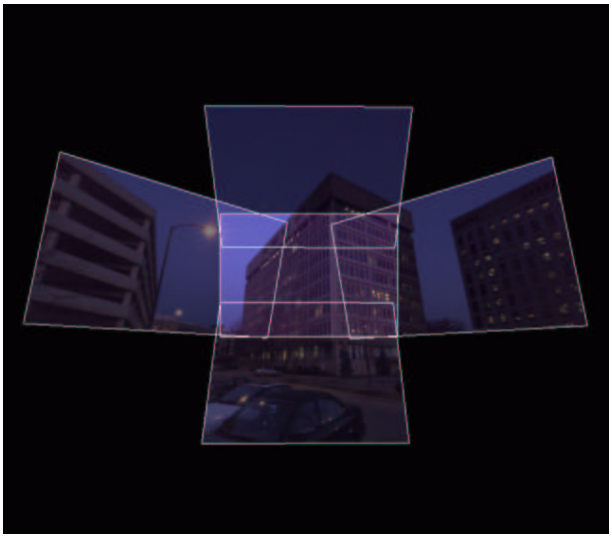

Figure 6: Left : view from the camera optical center. Right : combining raw images - Tech Square is visible at left; Draper Labs at right.

corresponds to phi (azimuth, from 0 to 360 degrees), and whose vertical coordinate corresponds to theta (altitude, from -90 to +90 degrees). Figures 8 and 9 show such "spherical" images, (again) composited from 47 source images.

Each spherical mosaic represents roughly 1012 x 1524 pixels/image x 47 images, or just under 75 million pixels. Thus, a square texture representing the mosaic would have a resolution of about 8,500 x 8,500 pixels. For technical reasons (basically, so the mosaics can serve well as texture maps inside OpenGL), both their horizontal and vertical dimensions must be powers of two. The closest matching powers of two which produce reasonable aspect ratios (remember, a hemisphere is 360 degrees of azimuth and 90 degrees of altitude) are 8,192 x 4,096 pixels. The mosaics pictured above are decimated; they have a resolution of 512 x 256 pixels, or 1/16th x 1/32nd = 1/512th of the information of the full-scale mosaics ! We have also used the Argus and its mosaicing algorithm to

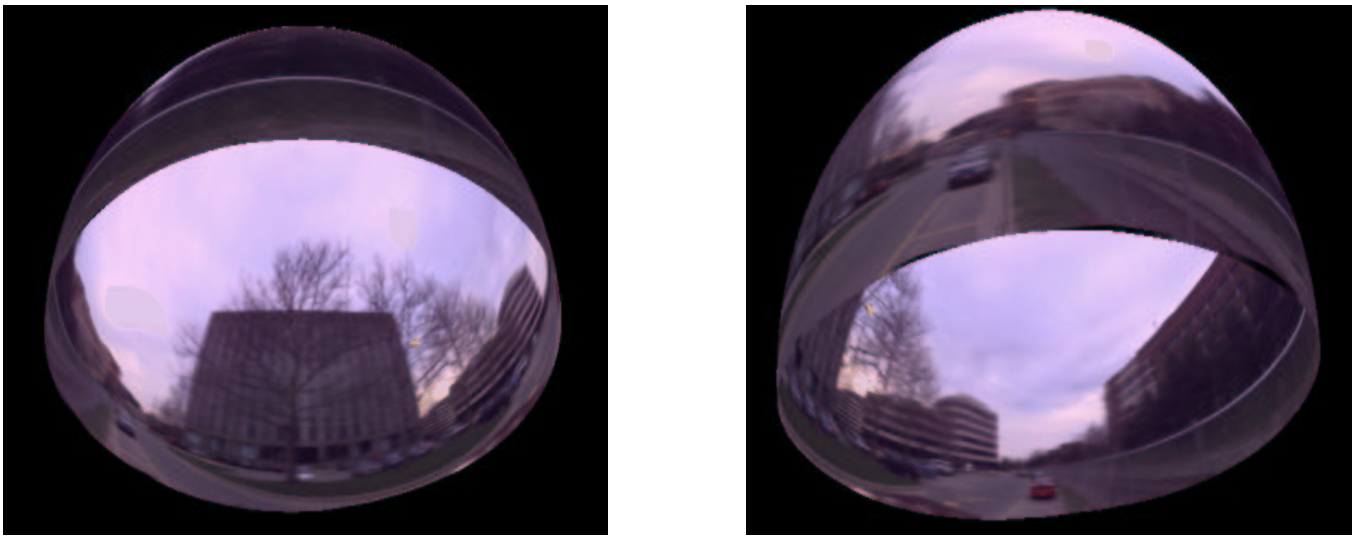

Figure 7: Several views of a spherical mosaic

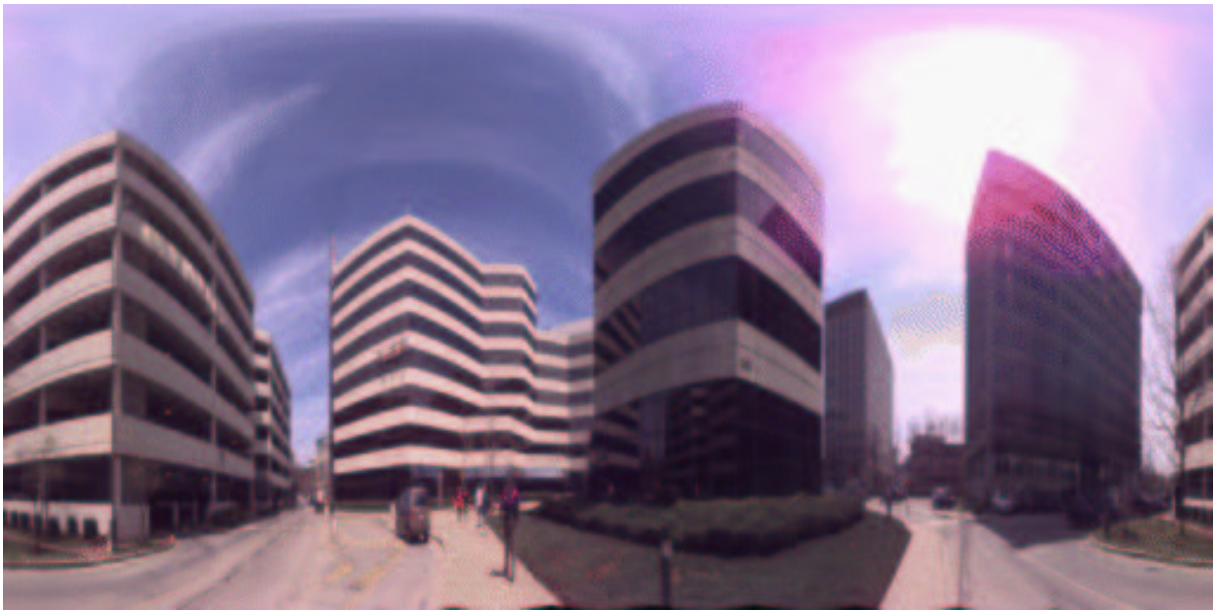

Figure 8: Mosaic spherical coordinate representation - X axis : azimuth; Y-axis : altitude

acquire some truly high-resolution (say, 8 gigapixel = 128K x 64K) mosaics of interesting interior spaces around MIT, such as Lobby 7. Producing spherical mosaics yields two significant advantages. First, we can treat the mosaics as rigid, "virtual" images with an effective field of view much higher than that of a raw image. In ensuing optimizations, this rigid virtual image can be manipulated as one entity – reducing, by more than a factor of fifty, the number of free variables to be optimized. Second, the wide effective field of view allows much more robust estimation of vanishing points and translation direction than does a single image. (In typical "frame" images, there is a fundamental ambiguity between small rotations and small translations. This ambiguity does not occur for super-hemispherical images).

Figure 9: Another image taken at twilight

1.3 The pipeline

This section presents an overview of the City Scanning Project pipeline. The system is composed of a set of programs - each one being attributed to a very specific task. Figure 10 represents the whole pipeline for 3D-geometry reconstruction. This pipeline itself is included in a complete system including sensor deployement as well as visualization and simulation applications. Here are some details about each step :

- undistort : the camera focale length may vary between 20 and 28 mm, which means that it is an "abnormal" lens - 50 mm is a typical normal length. This means that the image is distorted : lines get curved. To remove this distorsion, we use a program called undistort which uses the cameras intrinsic parameters to undistort the images.

- Mosaicing: in this step, we convert the set of raw images into spherical images, as described in the precedent section. Also, mosaicing is used to refine camera intrinsic parameters by rotating and translating the camera position. This step is very important, insofar as bad camera positions can lead to a bad edge correlation.

- Edge detection : as described in section 1.4.1, the edges are detected in the cubical images. These edges are used in the next step to run reconstruction.

- Reconstruction : in this step, we extract the 3D geometry of the environment from the edge detection, using a space sweep technique.

- Texture : finally, texture is computed for each vertical facade to produce a textured 3D model of the environment.

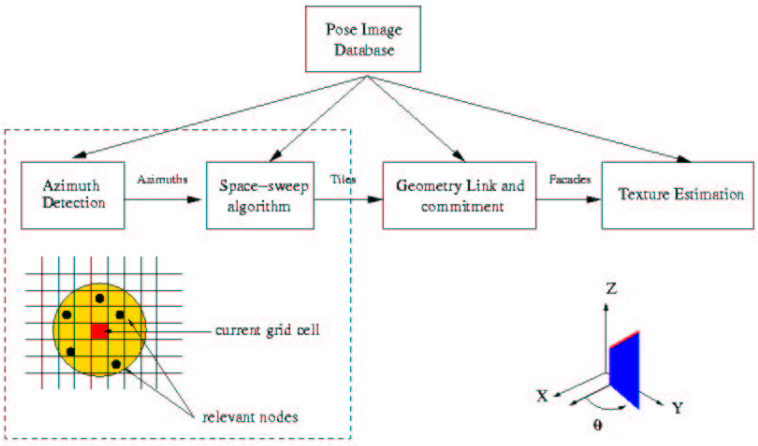

Figure 10: Reconstruction pipeline

1.4 Geometrical reconstruction

1.4.1 Space-sweep algorithm

The main part of our work has been done on reconstruction. This section describes how reconstruction basically works. As we said, we use a space sweep technique. In a first step, each horizontal edge is assigned an azimuth theta using the method described in section 1.1. Figure 11 shows the results of applying this technique to two nodes. Vertical edges are colored blue, and others edges are colored with the tile normal derived from their azimuth (e.g red is [1,0,0], green is [0,1,0], etc...)

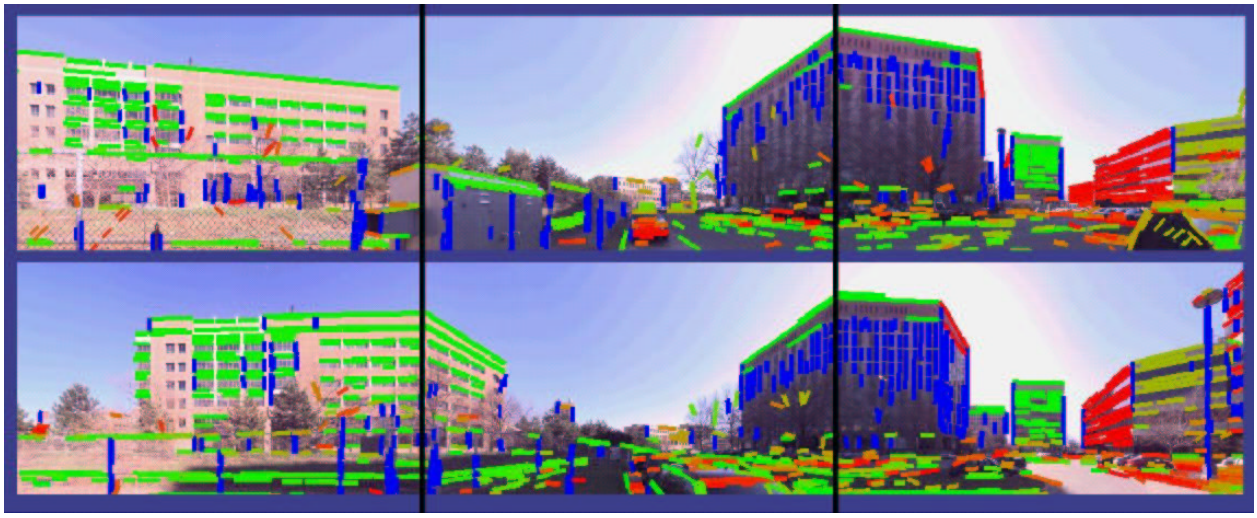

Figure 11: Edge detection from two nodes - edges are colored according to their azimuth

Now the question is : how can the camera decide if the edge is horizontal or not ? The idea is that an horizontal edges is always assigned the same azimuth, whereas azimuths of non-horizontal edges will vary with camera position. The use of statistical tools, such as histograms, allows to determine true azimuths. The algorithm picks a dominant azimuth from each node, then reports azimuths that are supported by several nodes. These azimuths are then verified by the space-sweep algorithm.

Basically, the space-sweep algorithm sweeps through the space for each probable azimuth. At each step, a correlation function based on binary search trees ([7]) is calculated. This correlation function is made so that it reaches a peak when the sweeping plane corresponds to the right position of the 3D facade. For more information concerning this function, see [8]. The area of interest is divided into cells. For each cell, the algorithm is called, and sweeps through the cell (figure 12).

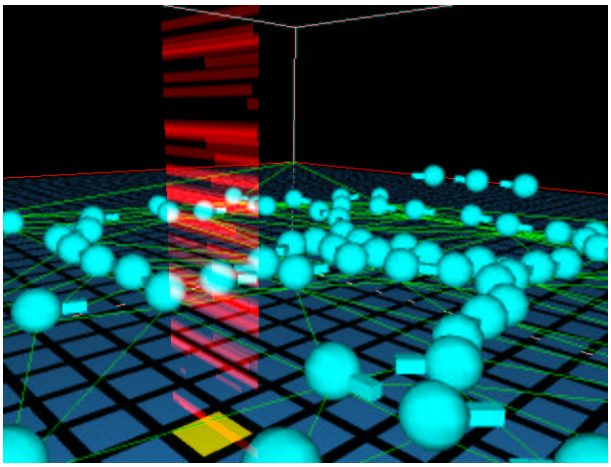

Figure 12: Sweeping plane through a spatial region of the dataset (in yellow) - blue domes represent camera positions. Red line segments are the projection of edges onto a virtual vertical plane. Color gets brighter when edges overlap.

Once the whole area has been processed, we apply some post-processing to eliminate spurious facades (figure 13) and link tiles from different grid cells to form complete facades (figure 14). Figure 15 shows one way in which a spurious facade can result by the interaction between unrelated facades of the same azimuths.

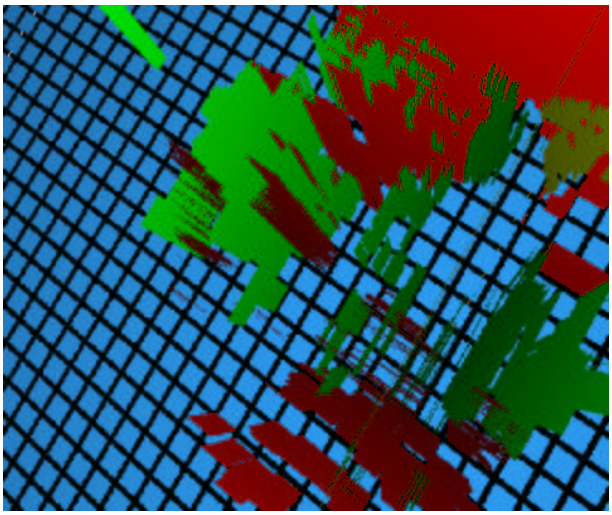

Figure 13: Reconstruction of 3D geometry in Technology Square

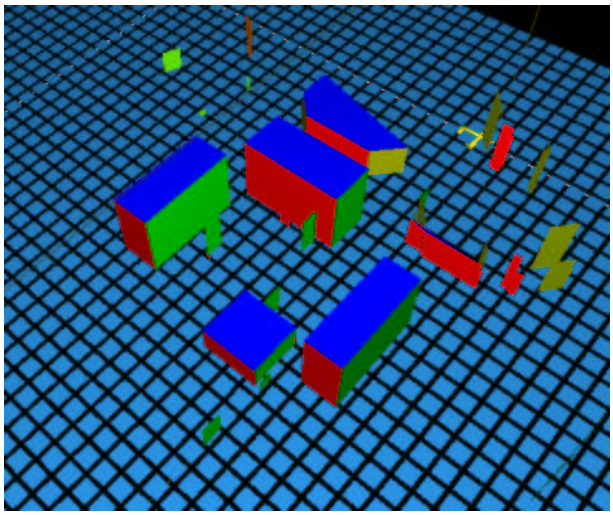

Figure 14: Tech Square geometry after connecting tiles

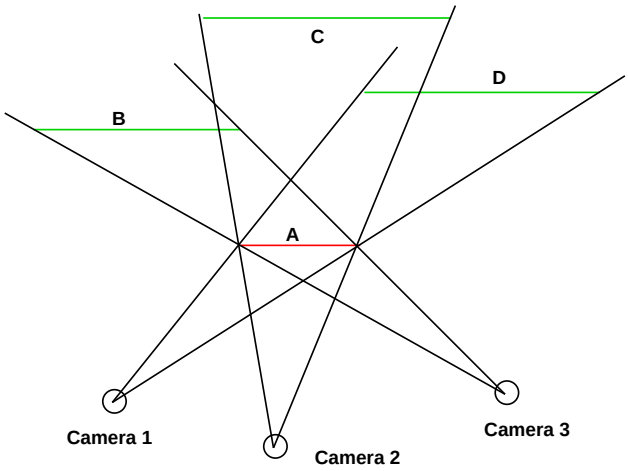

Figure 15: Some situations may create spurious facades

It might happen that, for several configurations, spurious facades appear. The figure opposite represents such a situation. Facades B, C and D are real, but the program may assign them a fourth facade (facade A). This case happens for example when a tree creates a lot of edges. In that case, the program may try to match the edges of the tree with other edges (such as windows edges) wich leads to spurious facades. One way to solve this problem is to enhance the quality of the 3-D model (e.g using priority ordering-based heuristic, see [8]). Another way would be to improve edge detection by filtering out edges in high-frequency areas of the image.

1.4.2 First results

This method has been applied first in Technology Square, on the east corner of MIT campus. 81 nodes have been used, i.e approximatly 4000 images. The area is 250,000 square meters. The facade extraction took about seven hours on a Silicon Graphics O2 workstation with one R10000 processor, most of which was spent in the space-sweep algorithm. The experiment showed that a facade must be observed by at least five nodes to be successfully extracted. Figure 16 shows the result of reconstruction. The right image is a textured model of Technology Square. The four buildings have been reconstructed as well as several facades around the area.

However, the reconstruction is not perfect. We have noticed several random behaviours and there are still some problems to be solved. In the next section, we will demonstrate the improvement we have brought to the reconstruction code.

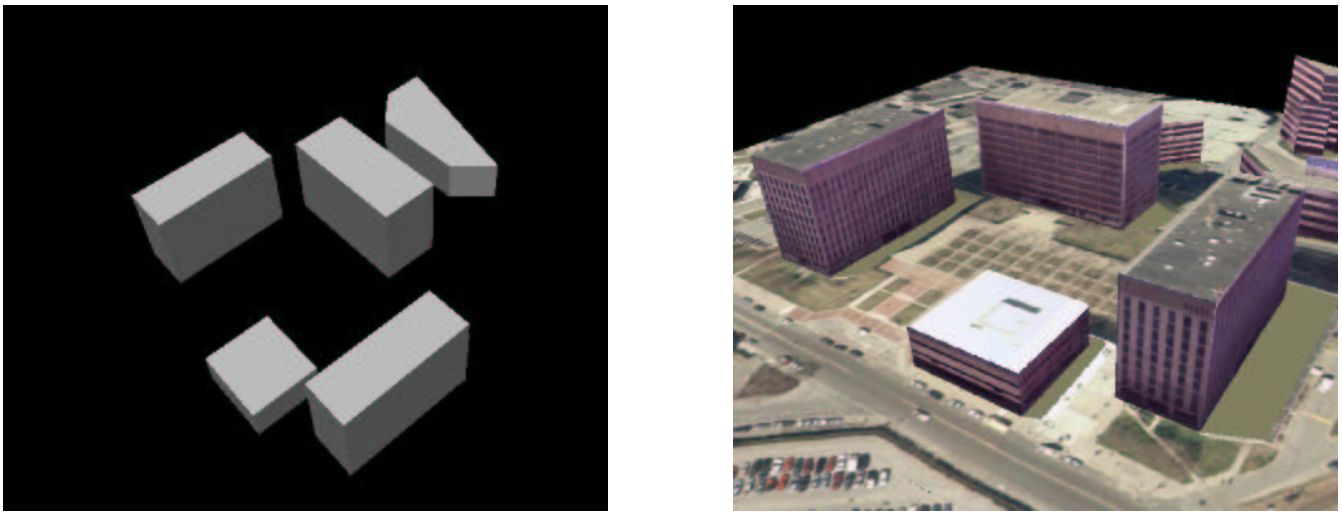

Figure 16: Reconstruction of Technology Square. Computation time : 7 hours on a SGI R10000 monoprocessor.Left : polygons. Right : textured model.

2 Code validation

In this section, we describe how to improve the reliability of the reconstruction code. The reconstruction deals with a huge database, and it is important to make sure that there is no bug in the code. If the program switches two polygons, this may have bad consequences - such as the creation of a spurious facade - and such an error is quite difficult to detect. If something is going wrong with the program, either it comes from an external reason, and we have to prove it, either it comes from an internal reason, and we have to solve it. Here we show two examples to illustrate these two possibilities. In the first example, we show in which way a bad edge detection can have an impact on the result of reconstruction (external problem). In the second example, we demonstrate a new feature that has been implemented to improve the result of reconstruction and get rid of some spurious facades (internal problem). The two examples take place in Technology Square.

2.1 Edge detection

The reconstruction program takes as input the edge files containing the edges of each image for each node. Each edge is attributed a weight according to its length : the longer an edge is, the bigger its weight is. The edge detection is made so that it takes into account all the edges of the image : it does not proceed to any filter and a high number of edges are irrelevant. An example of possible improvement is shown in section 5.1. In addition to this problem, edge detection is highly sensitive to shades : if the facade of a building is partially in the shade, the edge detection will be less efficient in the shaded area. Figure 17 illustrates this phenomenon.

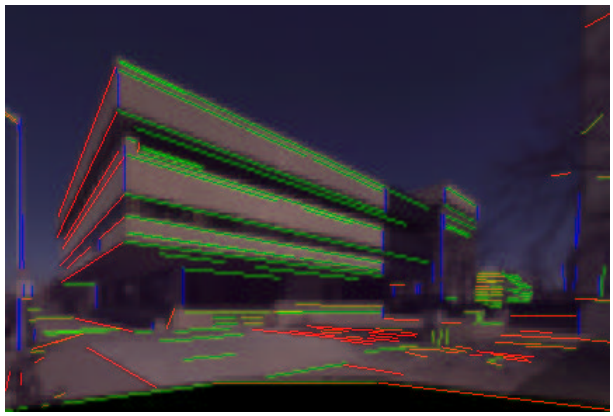

Figure 17: Edge detection on Polaroid building in Technology Square - Edges have not been correctly detected in the shade area.

However, the edge detection is an important step of the reconstruction, and a bad edge detection can produce a problem in the result of reconstruction. Figure 18 shows the result of reconstruction with the edges seen on figure 17. As we can see, the eastern facade of the Polaroid building has not been retrieved.

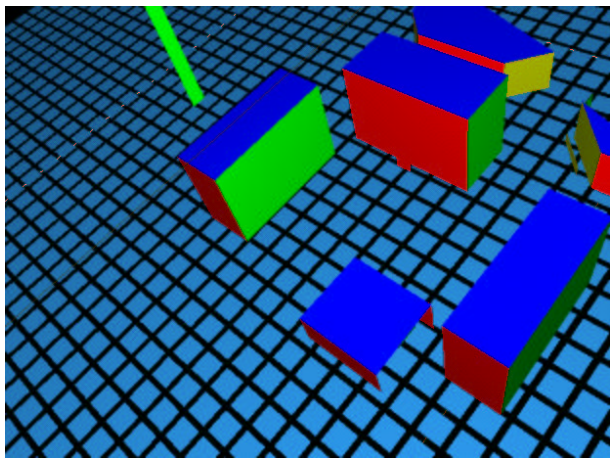

Figure 18: Reconstruction of Tech Square based on automatic edges. Eastern facade of Polaroid building is missing.

To prove that this problem is due to a bad edge detection, we have manually generated the edges for the nodes which have a view on this facade - 5 nodes in total. Figure 19 shows these edges. We have carefully detected the edges on the eastern facade. The result of reconstruction is represented on figure 20. The eastern facade has been successfully retrieved. As a conclusion, we can say that edge detection has a strong impact on the result of reconstruction. We think that this result is interesting, insofar as it specifies a constraint for the observation : each facade has to be observed at least six times, and the smaller it is, the more it needs to be observed. This information could be used to create a displacement algorithm for Argus for example - at the moment, Argus is displaced manually.

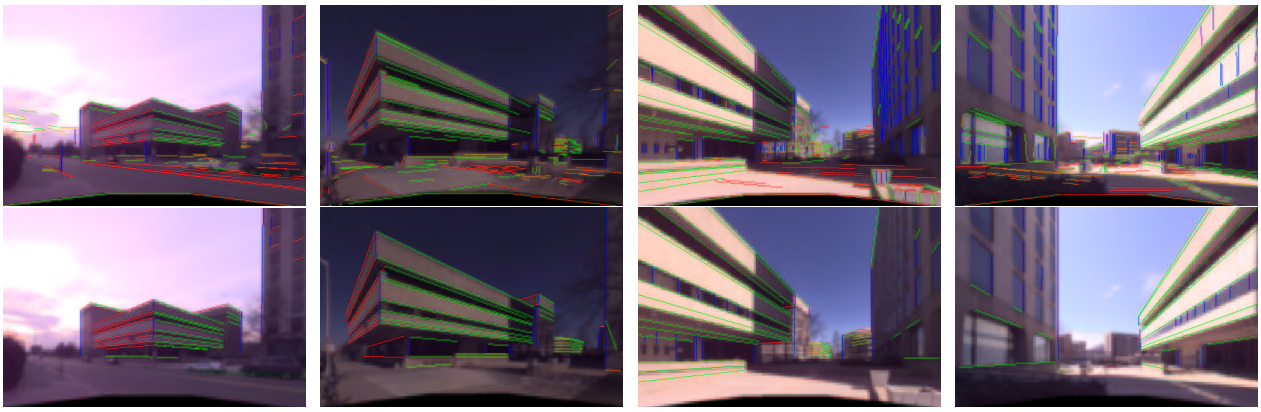

Figure 19: Edge detection for several nodes around Polaroid building. Top : automatic edges. Bottom : Manually generated edges using Gimp.

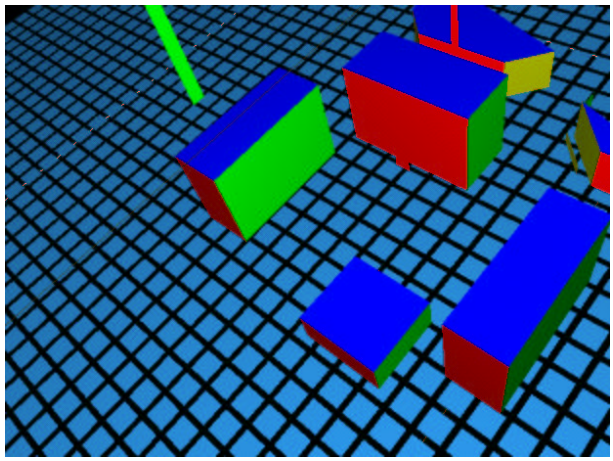

Figure 20: Reconstruction of Tech Square based on manually modified edges. Eastern facade of Polaroid building has been retrieved.

2.2 Polygons filtering

Here we discuss the case of false facades in reconstruction. Indeed, the results of reconstruction sometimes present several false facades. However, each facade is supported by a number of horizontal edges. Among the false facades, some of them are easily detectable : they are not supported by edges near the ground. This means that these facades have been generated by some spurious edges far away in the sky. We have implemented a filter to detect such facades. Figure 21 shows the result of this filter. As we can see, some of the spurious facades have been removed. The criteria used in the filter is not specific to the given dataset. We consider that a polygon is not well supported if there is no edges below four times the length of a grid cell - in the example of figure 21, the grid cell length is 500 feet. This criteria is arbitrary and should be tested on other datasets, but we believe that it is efficient.

Figure 21: Reconstruction on Tech Square without filter (left) and with filter (right). Several spurious facades have been removed.

3 Parallel algorithms

3.1 Why parallelize ?

The question of parallelizing or not parallelizing is not obvious. First, parallelizing a system might be a time-consuming task, especially if the program has not been implemented in a parallelizable way. It is then important to parallelize a code only if it is rewarding in term of computation time. Second, parallelizing a code always implies idle time and communication time. This slows down the system and the speed-up is rarely linear. In some cases, these idle and communication time can dramatically slow down the system, and using a parallel architecture is not rewarding anymore. Here is a short example to illustrate our argument. Let us assume that we have a 54-card game to sort. It is probable that distributing the task to 4 or 5 persons will be rewarding, since they can communicate quite easily. However, using 54 or more persons would be dramatic, since each person would have nothing to do, except trying to communicate with the other persons.

In the case of the City Scanning Project however, it is clear that parallelizing is rewarding. Running the reconstruction code on Technology Square dataset takes almost 7 hours. In addition, since the area of interest is divided into grid cells, a part of the work has already been done, given that the process of a cell is independent from the one of its neighbors, at least for the most time-consuming part of the pipeline.

Parallelizing reconstruction code not only allows to go faster. It is also a progress in a sense that it is now possible to think about a real-time reconstruction code which could be used as a feed-back for Argus motion. We can imagine a system in which Argus sends the images on the flow to the computation center and receives information which will have an impact on its future motion. For example, if Argus has detected a building on its left, it will prevent it from moving toward the left and it will also make it concentrate on this building.

3.2 First method

As we said, the area of interest is divided into grid cells. We have verified that, concerning the most time-consuming part of the process, the process of a cell is independent of the process of the others. We can therefore deduce two different methods of parallelizing. The first method maps the grid cells onto the cluster of machines. We have implemented a method to compute the weight of each cell. Indeed, the computing time varies across the cells and assuming that this time is constant would lead to a bad distribution of tasks. We have also implemented a fast method to quickly determine the CPU power of a given machine. Given this two elements, we have implemented a program which maps the grid cells onto the cluster. This program is completely generic. Figure 22 shows the computing time in seconds with respect to the grid cells for the Technology Square dataset. As we can see, the distribution of time is far to be homogeneous.

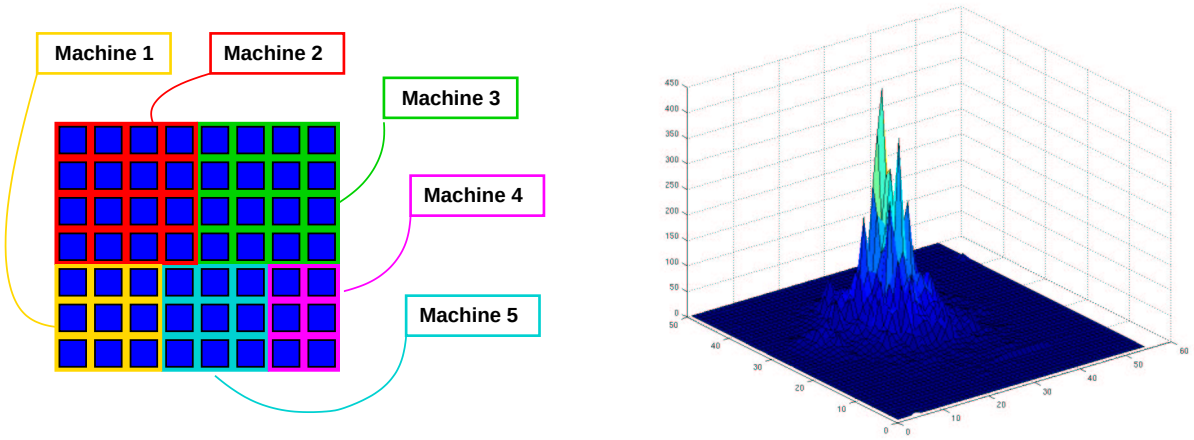

Figure 22: Left: mapping of the grid cells onto the machine cluster. Right : computing time on Technology Square with respect to the cells. Unit : seconds. The distribution is not homogeneous on the grid.

Our method consists in dividing the grid into rectangles (figure 22) of same weight in term of computation time. The number of rectangles is equal to the sum of all the CPU units available. This number is different from the number of machines. For instance, if we have three machines (machine 1, machine 2 and machine 3) and if machine 1 has 1 CPU, machine 2 has 1 and machine 3 has 4, then the number of rectangles is 6. We then distribute the rectangles to the machines with respect to the number of CPU units they have. In our example, machines 1 and 2 would be attributed 1 rectangle each, and machine 3 would be atributed 4 rectangles.

The advantage of this method is that it is flexible and quite easy to implement. Also it allows to have a first idea of the speed-up that we can expect. However, it implies a first initialization of the process to compute the weight of the cells and, as we will see in the next section, it is slower than the second method. Table 1 shows the computing time for each machine of the cluster. We have used 8 IRIX machines. The total computing time is 45 minutes. Comparing with the serial

Machine	Computing time (sec)
turpentine	2285
ray	2768
trace	2443
mosaic	2131
acetone	2711
hue	2342
panorama	1755
orange	1939

Table 1: Computing time for Technology Square reconstruction (in seconds).

Machine	Computing time (sec)
turpentine	10.96
ray	18.77
trace	17.77
mosaic	15.13
acetone	10.99
hue	18.98
panorama	10.87

Table 2: Computing time for the benchmark (in seconds)

version (7 hours), we have obtained a speed-up of 9. The distribution standard deviation is 3.4 %.

Some of the machine have several CPU units, which explains why the speed-up is higher than the number of machines. In addition, some of the machines have been upgraded with RAM memory, which makes them more efficient. Table 2 shows the result of a benchmark program for the different machines.[1].

3.3 Second method

Here we describe a faster and more flexible method. The idea is still to distribute the cells to the machines. However, this method dynamically maps the grid onto the cluster. Basically, the method is based on a master-worker model implemented with MPI. Each time a machine gets free, it receives a cell to process. As soon as it has finished processing the cell, it sends a message to the master in order to get a new task to do. Figure 23 represents the communication scheme of our method. Table 3 shows the computing time for each machine of the cluster.

Each line of the table correspond to a process. We have used the same cluster as in the first method. The total computation time is 30 minutes, including 3 minutes of communication and 6 minutes of idle time. The computation times in table 3 have been calculated using function MPI_Wtime(). They do not include communication time or idle time. The communication time has been calculated by running the program with nothing to do for each cell. The idle time has been calculated by substracting the communication time and the computation time to the total

[1]The benchmark was composed of two loops of float and integer computation as well as standard output

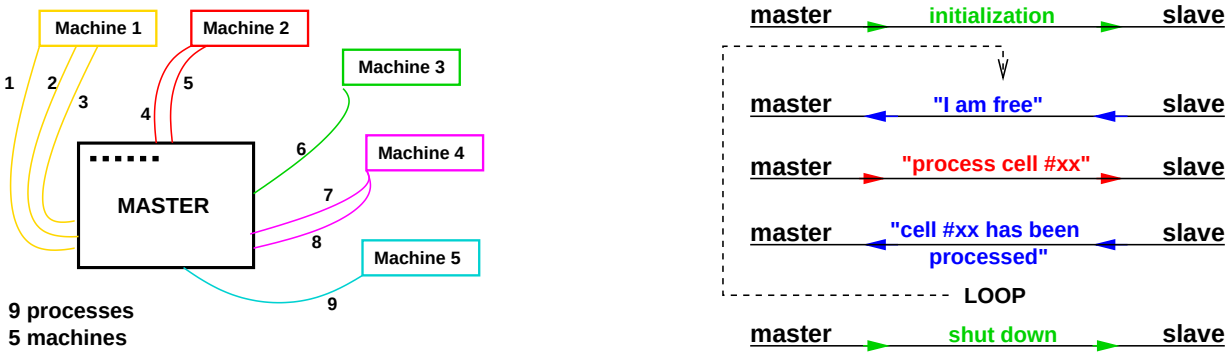

Figure 23: Left: master-worker model. Each machine can handle several processes. Right : communication scheme for the MPI implementation of the master-worker model on reconstruction program.

Machine	Computing time (sec)	Machine	Computing time (sec)
panorama	1246	acetone	1249
	1246		1239
	1248		1250
	1248		1238
	1250	mosaic	1255
	1240		1278
turpentine	1250	trace	1276
	1247		1276
	1243	orange	1277
	1233	hue	1268
ray	1230		

Table 3: Computing time in seconds for each process. Number of processes : 21.

processing time. One of the reason for which the idle time is so high is that each time a system call occurs on the master machine, the master proceeds to a new mapping of the processes.

The conclusion of this method is that it is much faster than the previous one : the speed-up is 15. It is also more flexible since it doesn't need any kind of static allocation at the beginning. The program only takes as input a list of machine.

3.4 Future improvements

Future improvements include a study of robustness. For now, if a machine crashes or is rebooted during the process, all the information is lost and the master will wait for its answer forever. It is possible to implement an acknowledgement system which would force the different machines to regularly send a message to the master. This system would allow to quickly detect if a machine is

down. The lost information could be then re-mapped on the lasting machines.

For now, the system does not test if the machine is reachable on the network and if the user is allowed to run a process on it. One could imagine a short initialization step during which this would be automatically tested.

Finally, it would be interesting to automatically determine the number of processes to be run on a machine with respect to the power of the machine. This would make the system even more flexible.

4 Camera position refinement

The City Scanning Project is based on geo-referenced images. Each time Argus takes a picture, it records its current GPS position as well as its intrinsic parameters and clip these information to the image. However, the accuracy of GPS positioning is 2-3 meters, and can be even worth in bad conditions. For this reason, the position of cameras have to be refined, so that we can reach an accuracy of several centimeters.

4.1 A bundle adjustment code

The bundle adjustment method is based on the idea that, if several cameras observe the same feature in the 3D world, it is possible to recover the relative position of the cameras given the orientation of the images. Figure 24 illustrates this method. Given a first approximate position of the cameras (basically, the GPS positions), the system tries to move the cameras around so that it minimizes the error on each 3D feature. Each feature is weighted with a magnitude corresponding to the weight of the edges from which the feature arises. Theoritically, if each camera observes 5 features, the position should be perfectly refined. In the real case, refinement is not so easy because we actually work on sphere images which are the result of a complex processing. Therefore, even if the method works well, we have encountered a few problems. The main one is that the system is extremely sensitive to the human error during the phase of selection of the features. In other words, if the user makes a mistake by selecting the wrong point on a sphere, it might prevent the system from refining well the cameras. Another problem is that it is sometimes difficult to find 5 relevant features for each camera. Indeed, the user can select a point only among a limited list of points - the edges intersections. And it sometimes happens that an interesting point cannot be selected because it is occluded by some tree for example.

Using this method, we have manually refined a subset of 27 nodes in the Ames dataset. We have then verified the improvement of the camera pose thanks to epipolar geometry. But the pose have not been improved enough to allow interesting results with reconstruction. However, the results of reconstruction have been improved, and we deeply believe that a correct set of camera pose would lead to a successful reconstruction of the area.

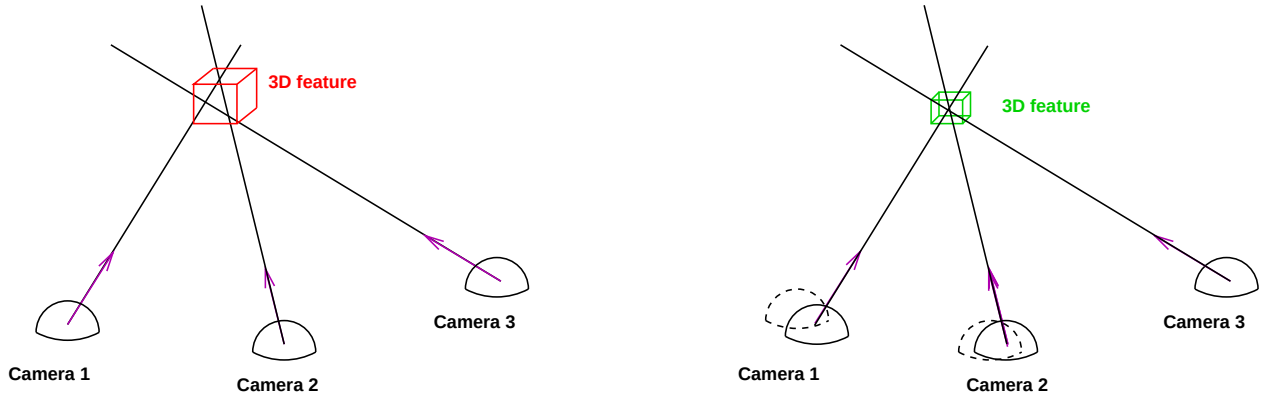

Figure 24: Bundle adjustment method : given the approximate orientation and translation of the camera (left), a new pose minimizing the error on the feature is determined (right). In the real case, several features are used in the same time.

4.2 Epipolar geometry tool

The City Scanning Project is also constituted of a tool called `carve` which allows to check the epipolar geometry of a set of cameras. If the relative camera pose is correct, then the epipolar geometry should also be correct. First, we select an image point for a given node. This point, combined with the camera center, defines a line in the 3D world. If we draw the projection of this line on the other nodes, we should see the line passing through the same point in the 3D world. Therefore, using 3D anchors like the corners of the buildings, we have been able to check the camera pose before and after refinement and to compare the results. It would be interesting to develop a tool to demonstrate the qualitative improvement of the camera pose after refinement. Figure 25 illustrates the idea of epipolar geometry. Figure 26 shows the epipolar geometry in Ames area before and after refinement.

5 A few more ideas

In this section, we describe several ideas which have emerged during our work on the City Scanning Project. The goal is to show how new ingredients could significantly improve several steps of the project.

5.1 High-frequency edge filtering

As we can see on figure 27, many edges are noisy on the images. Most of the time, these edges are due to high-frequency features such as trees. Our idea is to remove these noisy edges by detecting the areas of high-frequency on the images.

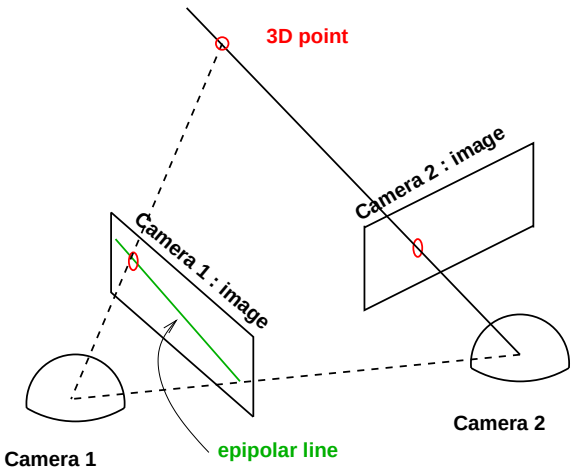

Figure 25: Epipolar geometry : to each 3D point seen from camera 2 corresponds a line on the image of camera 1. This line has to pass through the image of the 3D point on image plane of camera 1.

Figure 26: Epipolar geometry in Ames Street. Left : target point. Middle : epipolar line before refinement. Right : epipolar line after refinement. The epipolar geometry has been improved during refinement.

The first step is to convert the RGB image into a boolean (black and white) image. This image is then processed to detect the high-frequency area. To do so, we simply use a dilatation followed by an erosion. This method has been successfully used in other projects and presents the advantage of being a real-time process. The difference between the result of this process and the initial image produces a mask that can be used to lower the weight of the noisy edges. Figure 28 illustrates our method. As we can see, the tree has been perfectly detected. However, we can also see that the mask includes important edges such as window edges. A small improvement in the method could certainly help getting rid of that problem.

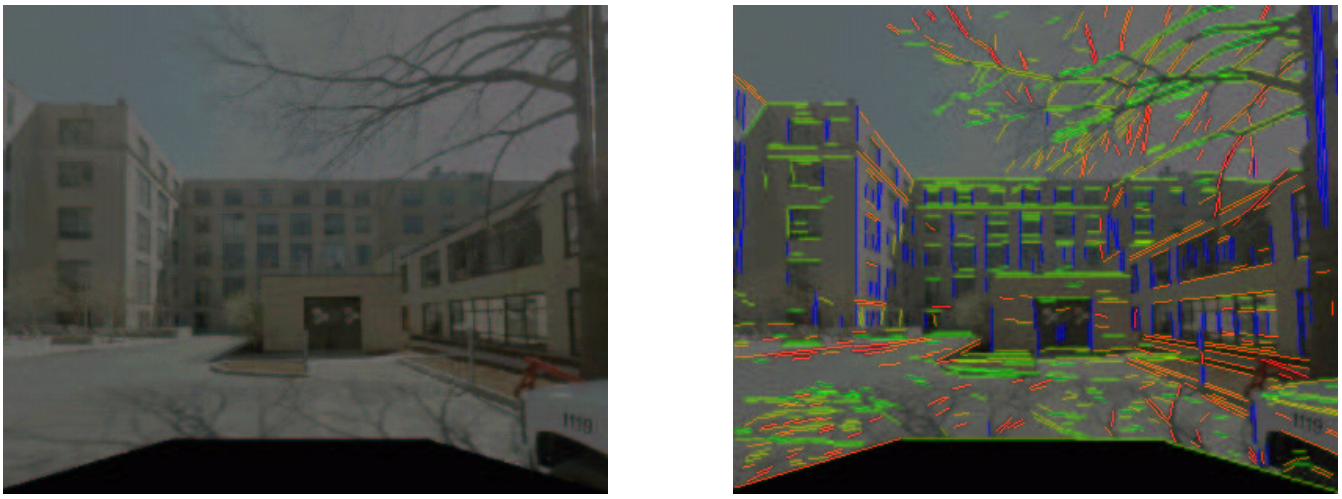

Figure 27: Edge detection on the image plane (Ames area dataset). Many edges are noisy (tree and its shade). Edges are colored according to their azimuth.

Figure 28: High-frequency filtering. Left : boolean image corresponding to image on figure 27 - Middle : high-frequency detection (dilatation-erosion) - Right : edge mask. Most of the noisy edges have been detected.

5.2 Improving Argus positioning

In section 4, we have seen that recovering the exact pose of the cameras might be extremely complicated. It is sometimes impossible to find the right camera pose using existing tools. In a general manner, recovering the camera pose from real-world imagery implies the use of complicated bundle adjustment code that can hardly be fully automated.

Therefore, we have thought about a system to determine the accurate camera pose while Argus is working. Indeed, we think that it is a shame to lose an information that can be gathered during the collect of data : why not accurately follow the motion of Argus ? The system would consist in several robots which would follow Argus as it moves. Our communication scheme is the following : at the beginning, Argus is stopped. It sends a message to the robots and moves. In the same time, the robots have recorded the motion of Argus. This record can either be done with laser, radio waves, imagery... The basic idea is to combine the information from the different robots to accu-

rately determine the new position of Argus. Of course, this implies that we also know the positions of all the robots. Then Argus stops and sends a new message to the robots. If Argus is occluded, the robots have to move so that they see Argus again. So far so good. Figure 29 illustrates the idea of our system.

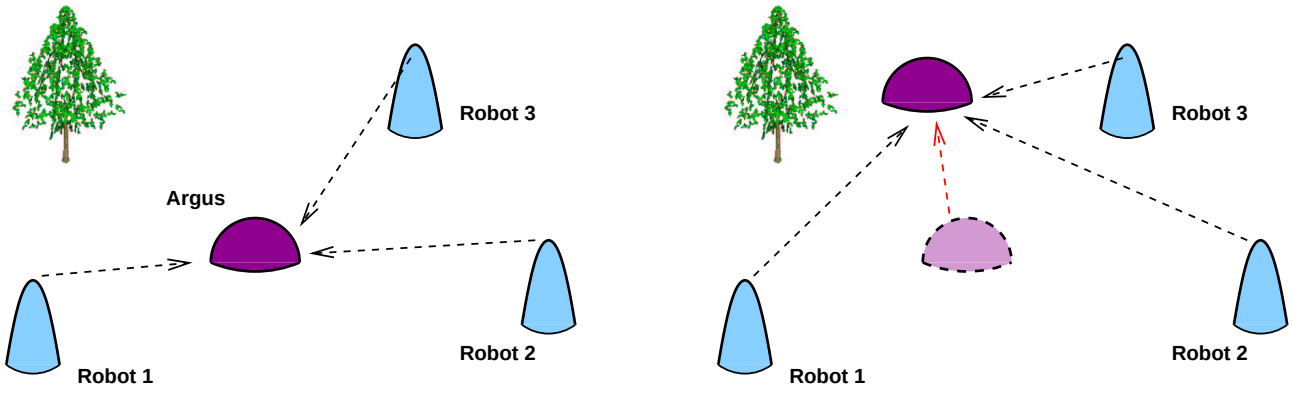

Figure 29: Several robots (here robots 1, 2 and 3) follow the motion of Argus and record it. This system can be used to recover an accurate set of pose for the images.

This idea would not be easy to implement, but it would give us a lot of information about the motion of Argus. We think that there is a way to determine the camera pose other than image processing.

5.3 Alternative sensors

The problems encountered with edge detection can be overcome with the use of alternative sensors such as IR or X-rays. Indeed, one of the main problem is that we detect noisy edges in the sky (clouds) or on the ground (trees). However, these features radiate very few heat in comparison with buildings. Figure 30 is a snapshot of a building in the IR wavelengths. We can cleary see that the sky and the features such as trees or road lights are black. Therefore, using this kind of images as a mask can be very efficient. The use of other wavelengths could be very interesting too.

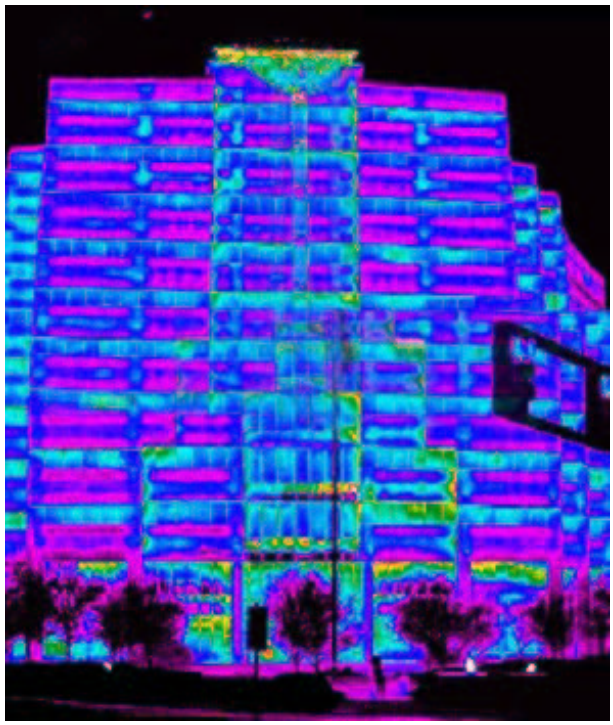

Figure 30: Copyright ©2002 : Infrared Services Inc. - A building seen in the IR wavelengths. Noisy features (trees, sky, road lights) are black. The black area could be used as a mask to filter out edges.

6 Conclusion

The main part of our work has been done on parallel algorithms. Even if the final implemented solution seems easy, many problems have been overcome first, including the validation of the code. Before running the reconstruction code, we had to check that each cell was independently processed, which was not the case in fact. We finally found a way to correct that. Thanks to a powerful IRIX cluster, we have obtained a significative speed-up of 15 and compared two different methods of parallel algorithms. The use of the 32-Linux machine cluster of the MIT Computer Graphics Group should allow even better results.

During this work, a lot of other ideas have emerged in front of several problems we have encountered : how can we improve camera pose refinement ? How can we improve the edge detection ? How can the final processing of polygons after reconstruction be more efficient ? These are some of the numerous questions we have tried to answer. This lead us to work with several programs of the project (refinement, epipolar geometry, reconstruction). Some of the problems have been solved. Other still need some work and, because of a lack of time, we could only propose solutions to them.

7 Acknowledgements

I first would like to thank Prof. Seth Teller for having offered me the wonderful opportunity to work at MIT. I received full support and complete trust during my stay here, and I have to admit that I never worked with such a feeling of freedom. My stay here has been rewarding at all aspects. Also I would like to thank Neel Master who worked with me during the first two months of my stay, providing me with a full understanding and an extremely strong technical support. I learned a lot about the project (and other stuff) thanks to him. Thank you also to all the graphics people who made my stay here so much agreable. I would like to adress a special thank to the french company, in order of appearance : Alexis, Fredo, Yann and Stefane. Finally I would like to thank Francoise Levy-dit-Vehel (ENSTA) and Prof. Laurent Li (LMD, Paris Jussieu) for having so kindly supported me during my application at MIT.

References

[1] Teller S., MIT City Scanning Project : Fully Automated Model Acquisition in Urban Areas. Available at : http://city.lcs.mit.edu//city.html

[2] Teller S., M. Antone, Automatic Recovery of Relative Camera Rotations in Urban Scenes, *Proc. CVPR, pp. II-282–289*, June 2000.

[3] Bosse M., D. de Couto & S. Teller., Eyes of Argus : Georeferenced imagery in urban environments, *GPS World*, April 2000, pp 20-30.

[4] Teller S., with Matthew Antone, Zachary Bodnar, Michael Bosse, Satyan Coorg, Manish Jethwa, and Neel Master, Calibrated, Registered Images of an Extended Urban Area, *to appear in Proc. IEEE CVPR*, December 2001.

[5] Bodnar Z., A Web Interface for a Large Calibrated Image Dataset, *AUP Report MIT*, May 2001.

[6] Wang X., S. Totaro, F. Taillandier, A. R. Hanson, S. Teller, Recovering Facade Texture and Microstructure from Real-World Images, *draft paper*, 2002.

[7] Bentley JL, Multidimensional Binary Search Trees Used for Associative Searching, *Communications of the ACM*, Sept. 1975, vol. 18, number 9.

[8] Coorg S., S. Teller, Automatic Extraction of Textured Vertical Facades from Pose Imagery, *MIT Computer Graphics Group*, Jan. 1998.

[9] Antone M., Robust Camera Pose Recovery Using Stochastic Geometry, *PhD thesis, MIT*, 2001.

[10] Coorg S., Pose Imagery and Automated Three-Dimensional Modeling of Urban Environments, *PhD thesis, MIT*, 1998.

8 Appendix : the MIT Computer Graphics Group

The City Scanning Project has been developed at MIT Computer Graphics Group, Cambridge MA. In this section, we give a brief description of the lab and its projects.

8.1 The Massachusetts Institute of Technology

Complete information about MIT and its history can be found at : http://www.mit.edu

Mission

The mission of MIT is to advance knowledge and educate students in science, technology, and other areas of scholarship that will best serve the nation and the world in the 21st century.

The Institute is committed to generating, disseminating, and preserving knowledge, and to working with others to bring this knowledge to bear on the world's great challenges. MIT is dedicated to providing its students with an education that combines rigorous academic study and the excitement of discovery with the support and intellectual stimulation of a diverse campus community. We seek to develop in each member of the MIT community the ability and passion to work wisely, creatively, and effectively for the betterment of humankind.

About MIT

Massachusetts Institute of Technology – a coeducational, privately endowed research university – is dedicated to advancing knowledge and educating students in science, technology, and other areas of scholarship that will best serve the nation and the world in the 21st century. The Institute has more than 900 faculty and nearly 10,000 undergraduate and graduate students, and is organized into five Schools – Architecture and Planning, Engineering, Humanities, Arts, and Social Sciences, Management, and Science – and the Whitaker College of Health Sciences and Technology. Within these are twenty-seven degree-granting departments, programs, and divisions. In addition, a great deal of research and teaching takes place in interdisciplinary programs, laboratories, and centers whose work extends beyond traditional departmental boundaries. The board of trustees, known as the Corporation, consists of about 75 leaders in higher education, business and industry, science, engineering and other professions.

8.2 The MIT Lab For Computer Science, LCS

Complete information about MIT LCS and its history can be found at : http://www.lcs.mit.edu.

About the Lab

The MIT Laboratory for Computer Science (LCS) is an interdepartmental laboratory whose principal goal is research in computer science and engineering. It is dedicated to the invention, development and understanding of information technologies which are expected to drive substantial technical and socio-economic change.

LCS research has spawned over three dozen companies, including 3Com Corporation, Cirrus Logic, Inc., Lotus Development Corporation, Open Market, Inc., RSA Data Security, Inc., and Akamai Technologies, Inc. The Laboratory hosts the USA headquarters of the World Wide Web Consortium, an open forum of companies and organizations with the mission to lead the Web to its full potential.

Currently, LCS is focusing its research on the architectures of tomorrow's information infrastructures. In the interest of making computers more efficient and easier to use, LCS researchers are putting great effort into human-machine communication via speech understanding; designing new computers, operating systems, and communications architectures for a networked world. In addition, LCS recently announced the launching of the Oxygen project, an integrated collection of eight new technologies: handhelds, wall and trunk computers, a novel net, built-in speech understanding, knowledge access, collaboration, automation and customization.

Organization

Most members of LCS are affiliated with either the Department of Electrical Engineering and Computer Science (EECS) or the Department of Mathematics at MIT. The Lab has 65 faculty and senior research staff members, about 50 visiting faculty members, postdoctoral students, and research affiliates, and 180 graduate students. 100 undergraduates, working under MIT's Undergraduate Research Opportunity Program, also are intimately involved in LCS advanced research projects. Victor Zue is the Director of LCS, Anant Agarwal and Chris Terman are serving as Associate Directors.

8.3 The MIT Computer Graphics Group

MIT Computer Graphics Group is a research group within the MIT Lab for Computer Science. Complete information can be found at : http://graphics.lcs.mit.edu

The MIT Computer Graphics Group was founded a dozen of years ago by four professors wanting to gather their resources and knowledge. It houses several research groups working on the latest graphics technology. Current research projects include : Acoustic Design and Modeling, Motion Capture, Synthesis and Analysis, Weathering and Surface Appearance, Projective Drawing, Image-Based Modeling and Photo Editing, ... The lab has 4 faculty members and about 20 graduate students.

8.4 The City Scanning Project

The City Scanning Project began approximatly five years ago. Its principal investigator is Prof. Seth Teller. About 10 staff and students have worked on it during the past several years. The major funders are Intel Corporation, Lincoln Laboratories' Advanced Concepts Committee, DARPA, and ONR MURI. The web page of the project is : http://city.lcs.mit.edu

8.5 My work at MIT

Working at MIT Computer Graphics Group has been a wonderful experience to me. I have been working in excellent conditions with a crowd of interesting people. The group has always be an extremely warm place to live, providing me with a strong technical support as well as a lot of experience to share. People really do things well at MIT.

In a general manner, MIT is an amazing place to live : there's plenty of things to do there after work. MIT owns a dozen of huge libraries, including a music library providing scores for free. Most of the on-campus residences have a free piano - mine had a Steinway. There's something happening every day at MIT : undergrade and grade students organize social events, theater plays, movies, exhibitions, concerts, dance shows... and it's all so cheap !

MIT is located right between Harvard and Boston. Harvard is a wonderful place with green areas, bookstores, and lots of activity. One can spend hours there just looking at the people passing by. Boston is another world. It's another kind of beauty. It's rich, modern, and quiet. Everything is much more expensive than in Cambridge - theaters and restaurants. It's a very warm place where people talk to you when you sit on a bench, and maybe that's the reason why people say Boston is a european city ! Quincy Market, Boston Common, Beacon Hill, Back Bay, Chinatown and South Station are the places that have played a role during my stay in Boston. I had to mention them.