Since the assumption is that the room is rectangular, we know that the top and bottom wall must be perpendicular to this left wall. The width of the top and bottom walls are unknown, but their widths should be equal because the wall on the right side is parallel to the left wall.

Let us choose an arbitrary camera position on the arc and consider this position. Then, the corner-to-corner angles α_1 and α_2 define the direction of two rays q_1, q_2 emanating from the camera position in the direction of the room corners (Fig. 14.11). These rays intersect the top and the bottom walls in a distance w_t and w_b from the left wall, respectively. Because we know that the top and bottom wall should have equal length, w_t should equal w_b. However, if we have chosen the wrong camera position on the circular arc, this will not be true.

Notice that if we move the camera upwards along the arc, the top intersection point moves to the left (w_t decreases), while the bottom intersection point moves to the right (w_b increases). To find the camera position for which $w_t = w_b$, we can exploit this behaviour by applying a binary search for the correct camera position. If $w_t > w_b$, the camera position is further to the top, while for $w_t < w_b$, the camera position is lower.

For some camera position, the ray direction of q_1 or q_2 becomes horizontal. For these positions (and the more extreme positions), there is no intersection of the rays with the top or bottom wall. These critical camera positions can be used to determine an initial interval of camera positions for the binary search. Starting the search with this interval not only reduces the number of iterations needed for the binary search, but it also removes the requirement to handle the special case in which the rays q_1, q_2 do not intersect the top or bottom walls.

14.4.3 Creating a virtual room visualization

When the sizes of the room walls and the camera position are known, we can create a virtual 3-D model of the room and generate textures for the room walls. To create the texture maps, we scan the 3-D wall plane with the desired resolution of the texture maps and we compute the respective pixel position in the panoramic image by using the inverse of Eq. (14.1).

This obtained 3-D room model is rendered using an OpenGL-based viewer application. The scene is built with the estimated camera position as the origin of the 3-D coordinate system. The user can control the rotation of the scene around the x and y axes, as well as the distance d of the camera to the origin. The viewing transform is set up as

$$\mathbf{p}' = \mathbf{K} \left[\mathbf{R_x} \mathbf{R_y} \, \mathbf{p} + \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} \right], \tag{14.3}$$

where $\mathbf{p}$ denotes the 3-D point position, $\mathbf{R_x}, \mathbf{R_y}$ are the rotation matrices and $\mathbf{K}$ is the perspective projection matrix. This particular sequence of transformations allows for a very intuitive navigation. When the distance of the camera to the origin is decreased, the program avoids d to become negative. This makes it very easy to place the camera at the position of the real camera (move forward until the virtual camera reaches the original camera position). From that position, the user views the scene just as if he would be at the camera position in the real world. Panning with the virtual camera at this special position gives exactly the output as displayed by popular viewers for panoramic images. The second useful viewing position is looking down on the complete room from above the scene, since this gives a quick overview of the general scene layout. An example visualization created with the described algorithm is depicted in Figure 14.8.

14.5 Reconstruction of floor plans

The reconstruction algorithm described in the previous section was limited to rectangular rooms. In this section, we extend this algorithm to enable it to reconstruct the geometry of arbitrary rooms. We keep the principle that corners are manually marked in panoramic images, and that the algorithm derives the camera position and wall sizes from the angles between room corners. The layout of the room walls is also specified by the user. For more complex rooms, it is often impossible to see all walls in only one image because of occlusions. In these cases, the algorithm uses several panoramic images captured from different positions.

14.5.1 Previous work

Several algorithms for 3-D reconstruction have been proposed. They can be coarsely divided into algorithms without pre-knowledge about the scene and algorithms making use of a scene model. Algorithms of the first class are usually very complex to implement [149] and they are probably not robust enough in cases of low-textured surfaces. Algorithms of the second class employ a complete geometric model of the object or scene and they only adapt the sizes based on the observed images. An algorithm of this second class is described in [37]. Another algorithm [168] considers specifically the reconstruction of room shapes from panoramic images. Compared to our algorithm, it supports more general geometries than a collection of walls, but compared to our proposal, it is more complex to implement and to use.

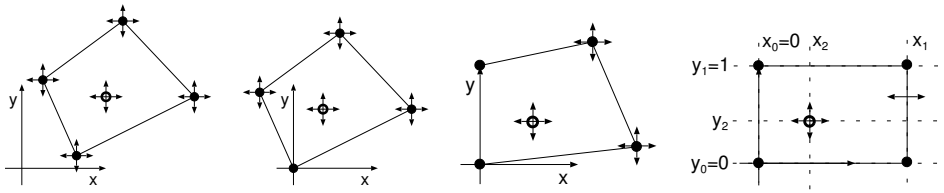
14.5.2 Reconstruction algorithm concept

Our floor plan reconstruction is based on the same user interaction as in the simpler rectangular room case. The user also marks the position of the room corners in the input image. However, while we previously only considered reconstruction from a single panoramic image, we now allow for an arbitrary number of panoramic images. This is necessary since many room corners can be occluded from some camera positions. The more panoramic images are used, the more information we have available for the reconstruction. The less pre-knowledge about the room geometries is available (non-perpendicular walls, unconnected free-standing walls), the more images are required.

The algorithm starts with an initial room configuration that defines the room layout (position of walls and constraints about perpendicular walls), but that does not yet include the correct wall sizes. For an example, see Fig. 14.13. This figure shows a user-supplied geometric room model, where the outline of the room is specified, but the correct wall sizes are still unknown. The basic principle of the algorithm is to compute the *corner-to-corner* angles from the current model and compare them with the measured angles. A gradient descent search is used to adapt the wall sizes such that the differences between angles in the model and the measured angles are as small as possible.

In the following, we describe the algorithm in four steps.

- **Section 14.5.3**. First, the parameterization of the model is constructed. Parameters are chosen such that hard constraints like perpendicular or parallel walls are enforced by the parameterization itself.
- **Section 14.5.4**. Second, we present the parameter-estimation algorithm. This step adapts the parameters such that the corner-to-corner angles in the model fit to the angles measured in the input images.
- **Section 14.5.5**. The convergence robustness depends on the definition how measured angles and angles in the model are compared. We compare an *inner-angle* definition with an *outer-angle* definition by examining the error function for local minima or plateaus, which decrease the robustness of the optimization.
- **Section 14.5.6**. Because the optimization is based on a gradient descent approach, a good initialization is required. The evaluation of the error function will show that local minima and plateau region can be avoided if the initialization satisfies some ordering conditions. In this last step, we explain how the initialization is obtained.



(a) Free position (10 parameters).

(b) Fixed position (8 parameters).

(c) Fixed rotation, size (6 parameters).

(d) Orthogonal geometry (3 parameters).

Figure 14.12: *By adding geometry constraints to remove unnecessary over-parameterization, we can reduce the number of parameters from 10 in the general case (a) to only 3 for a rectangular room (d). The free parameters are indicated with double arrows.*

14.5.3 Modeling the floor plan geometry

The floor plan reconstruction algorithm uses two types of information for the estimation:

- the angles between room corners measured from their position in the panoramic images, and
- the predefined geometrical layout of the room. This geometrical model includes the relative position of the walls, but not their size. The model also considers pre-knowledge about right angles between walls.

Let us first consider the number of degrees of freedom when estimating the floor plan geometry. A floor plan is parameterized by the 2-D positions of the room corners and the camera positions. The camera positions are required to carry out the texture mapping.

We start with a simple example of a rectangular room and one camera, which is similar to the special case that we considered in the previous section. This configuration gives 4×2 parameters for the room corners plus two parameters for the camera position (Fig. 14.12). However, the absolute placement of the room in our coordinate system is arbitrary and we can fix one corner to a predefined position, like $(0, 0)$. Moreover, we can fix the overall rotation angle of the floor plan, and as absolute size cannot be determined, we can also fix the length of one wall to, e.g., unity. The easiest way to do this is to fix the position of a second corner to, e.g., $(0, 1)$. In total, this reduces the number of degrees of freedoms by four.

The reduction from ten parameters to only six was obtained by eliminating superfluous degrees of freedom in the parameterization. On the

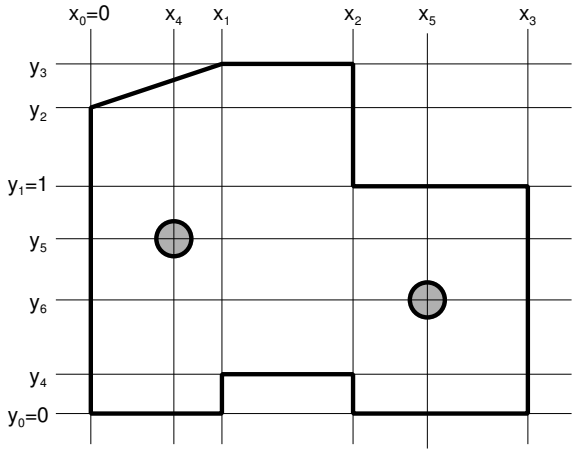


Figure 14.13: *Room corners are specified by coordinates x_i, y_i. Horizontal and vertical walls will reuse the same x_i or y_i coordinate for both corners. This implicitly encodes the pre-knowledge that these walls have to be horizontally or vertically aligned. Camera positions are assigned their own pair of x_i, y_i coordinates.*

other hand, we can add more pre-knowledge about the room geometry. For example, we can assume that the room shape is rectangular. This pre-knowledge can be expressed with three constraints, each forcing one wall to be perpendicular to another wall. These three constraints further reduce the number of free parameters from six to three, thereby making a reconstruction possible.

For our general floor plan reconstruction, we enforce the constraints for perpendicular walls implicitly through the parameterization. We normalize the rotation of the complete floor plan such that (most) walls will be aligned along the horizontal and vertical coordinate axes. Each wall that is aligned to the coordinate axes can be parameterized with only three parameters. For example, a vertical wall is parameterized by the two corner positions, but both positions share the same x coordinate. For the right wall in Fig. 14.12(d), we would get the corner positions (x_1, y_0) and (x_1, y_1).

Furthermore, we also add the normalization of the floor plan position and size as hard constraints in the parameterization. For this, we select one vertical wall and define one corner postition to $(x_0, y_0) = (0, 0)$ and the other corner position to $(x_0, y_1) = (0, 1)$. Note that using this parameterization for the rectangular-room case, only x_1 for the right wall position and x_2, y_2 for the camera location remain, so that we can compute these three free parameters from the three angle measurements.

A more complex example is depicted in Fig. 14.13. The room shape has eleven walls, but it is parameterized with only six free parameters $x_1, \ldots, x_3,\ y_2, \ldots, y_4$. Additionally, the two camera positions add four parameters $x_4, y_5,\ x_5, y_6$. From the image of the left camera, we can obtain nine angle measurements, since two of the walls are at least partly occluded. The right camera can contribute seven angle measurements. In total, we have 16 measurements for 10 parameters and the reconstruction is possible. Note that a reconstruction would also be possible with only the left camera. In this case, we would only have nine measurements, but also only eight parameters, since the position of the right camera is not included. On the other hand, a reconstruction from only the right camera is impossible, since we would have eight parameters to estimate from only seven measurements. Note that a sufficient number of measurements does not generally assure that the reconstruction is possible. This is the case when there are more measurements available than required for some walls and at the same time, too few measurements for other parts. However, in practice, this is rarely the case.

14.5.4 Estimating the floor plan parameters

The central task in the floor plan reconstruction is to estimate the model parameters based on the angle measurements that were taken from the panoramic images. The model parameters consist of the coordinates x_i, y_i of the wall corners and the camera positions. According to the geometric constraints, some of these coordinates can appear in the specification of several positions. All coordinate values that appear in the model are collected in a large parameter vector

$$\mathbf{v} = (x_0 = 0, x_1, x_2, x_3, \ldots,\ y_0 = 0, y_1 = 1, y_2, y_3, \ldots), \tag{14.4}$$

in which three entries are fixed (namely $x_0 = y_0 = 0$, and $y_1 = 1$) to remove the superfluous degrees of freedom. To find the corresponding coordinates for a position $\mathbf{p}_i$, we use two index sets m_i and n_i into the parameter vector $\mathbf{v}$ to define $\mathbf{p}_i = (x_{m_i}, y_{n_i})^\top$,

From the captured panoramic images, we obtain a set of angle measurements. Each measurement gives an angle $\alpha_{i,j,k}$ between corners $\mathbf{p}_i$ and $\mathbf{p}_j$, seen from camera position $\mathbf{p}_k$. We denote the set of available measurements as $\mathcal{M} = \{(i, j, k)\}$. Furthermore, we can compute angles $\beta_{i,j,k}$ that correspond to the measured angles from the geometric model as

$$\beta_{i,j,k} = \arccos \frac{\mathbf{d}_{ik}^\top \mathbf{d}_{jk}}{||\mathbf{d}_{ik}|| \cdot ||\mathbf{d}_{jk}||}, \tag{14.5}$$

where $\mathbf{d}_i = \mathbf{p}_i - \mathbf{p}_k$ and $\mathbf{d}_j = \mathbf{p}_j - \mathbf{p}_k$ are the vectors from the camera position k to the corners i and j. This equation defines $\beta_{i,j,k}$ as the inner angle between these vectors. Actually, we will shortly replace this definition with a slightly modified one that gives a better convergence behaviour. For an error-free ideal case, all measured angles $\alpha_{i,j,k}$ should equal the angles $\beta_{i,j,k}$, computed from the adapted model. Because of noisy measurements, these angles will not be exactly equal, and we define the total error of the floor plan model as

$$E = \sum_{(i,j,k)\in\mathcal{M}} |\beta_{i,j,k} - \alpha_{i,j,k}|, \tag{14.6}$$

which we minimize with a Quasi-Newton optimization. The convergence of this optimization depends on two factors: the smoothness of the cost function E, and the initialization. We discuss these two topics in the successive two sections.

14.5.5 Improving the convergence behaviour

When considering again the definition of $\beta_{i,j,k}$ from Eq. (14.5), we notice that the definition gives the non-oriented inner angle $\beta_{i,j,k} \in [0;\pi]$ between two vectors. During experimenting with this measure in the optimization process, we occasionally observed that the optimization did not converge. To see why the above angle definition can cause problems in the optimization process, consider the very simple case that there is only one camera which observes a single wall. Imagine that the camera is moved on a line perpendicular to the wall from one side of the wall through the wall to the other side. While approaching the wall, the angle $\beta_{i,j,k}$ increases to π and after crossing the wall, it decreases again. For this case, the cost function E is symmetric to the wall. In the optimization process, this has the disadvantage that there is a minimum of E on each side of the wall (Fig. 14.14).

To prevent this effect, we changed the definition of the angle $\beta_{i,j,k}$ to an oriented angle. We define the *oriented angle* $\beta'_{i,j,k}$ as the angle from corner $\mathbf{p}_i$ to corner $\mathbf{p}_j$, measured in counter-clock-wise direction (see Fig. 14.15). The orientation of the two corners is detected by computing the signed area spanned by the two vectors $\mathbf{d}_{ik}$ and $\mathbf{d}_{jk}$ from the camera to the wall corners. The signed area is obtained easily from the determinant of the matrix composed of these two vectors. Thus, we can compute the oriented angles as

$$\beta'_{i,j,k} = \begin{cases} \beta_{i,j,k} & \text{if } \det[\mathbf{d}_{ik} \,|\, \mathbf{d}_{jk}] \le 0, \\ 2\pi - \beta_{i,j,k} & \text{if } \det[\mathbf{d}_{ik} \,|\, \mathbf{d}_{jk}] > 0. \end{cases} \tag{14.7}$$

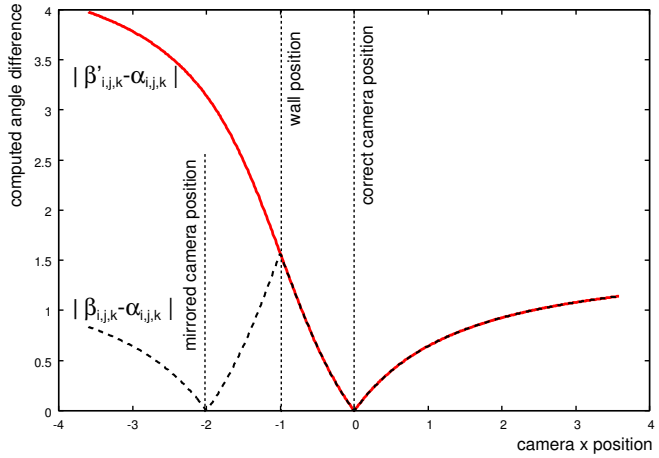
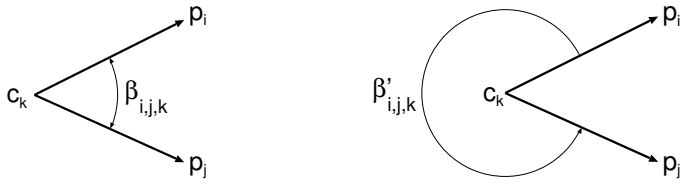


Figure 14.14: *With the non-oriented angle $\beta_{i,j,k}$, it cannot be distinguished on which side of a wall the camera is located. The oriented angle $\beta'_{i,j,k}$ has a single minimum at the correct side.*



(a) $\beta_{i,j,k}$ (b) $\beta'_{i,j,k}$

Figure 14.15: *Definition of angle differences. While $\beta_{i,j,k}$ is the inner angle between the two vectors (a), $\beta'_{i,j,k}$ is defined as the angle from $\mathbf{p}_i$ to $\mathbf{p}_j$ in counterclockwise direction (b).*

In comparison with the previous inner angle definition, the new oriented angle can distinguish between the camera being on the correct side of a wall and being on the backside of the wall. Using this angle definition, we obtain a clear minimum at the correct side of the wall, and the error E increases monotonically with increasing distance from the optimal position (Fig. 14.14).

Dependency of the model error on the camera position

Let us now examine a more complex case with a variable camera position in a rectangular room having fixed walls at $x = \pm 1$ and $y = \pm 1$. The error surface of E for the two angle definitions is depicted in Fig. 14.17. In the

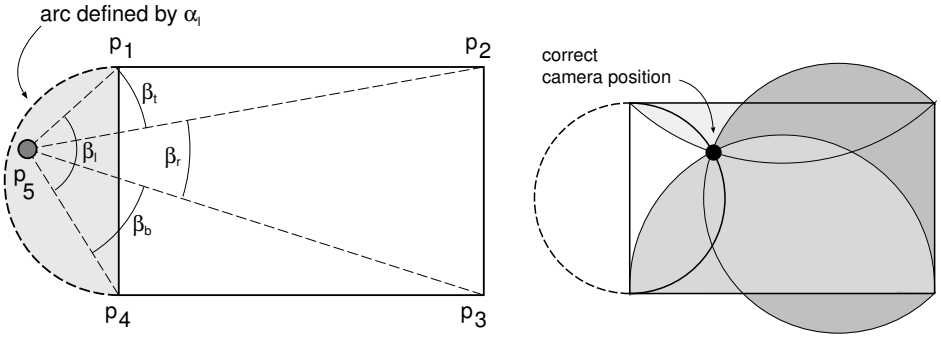


Figure 14.16: *Illustration of angles in a plateau area. (a) Moving the camera within the grey plateau area does not change the total cost E. (b) In each of the grey areas, $\alpha_i < \beta_i$. The symmetric areas, mirrored at the walls are not shown.*

case of the inner angle definition, we notice that there are *plateau* regions around the room walls with constant E. These areas impose difficulties, since the gradient based optimization can get stuck in this area.

To understand why these plateau regions exist, let us concentrate on one of these regions as depicted in Figure 14.16(a). The depicted area corresponds to the area outside of the room, for which $\alpha_l < \beta_l$, which means that the camera in the model is closer to the wall than in reality. For simplicity of notation, we use the notation $\alpha_l = \alpha_{1,4,5}, \beta_l = \beta_{1,4,5}$ as abbreviation for the angles corresponding to the left wall. We use similar abbreviations for the top (t), bottom (b), and right (r) walls. For the considered plateau area at the left wall, $\beta_r < \alpha_r, \beta_t < \alpha_t, \beta_b < \alpha_b$ as illustrated in Figure 14.16(b). If we compute the total angle error for all four walls as

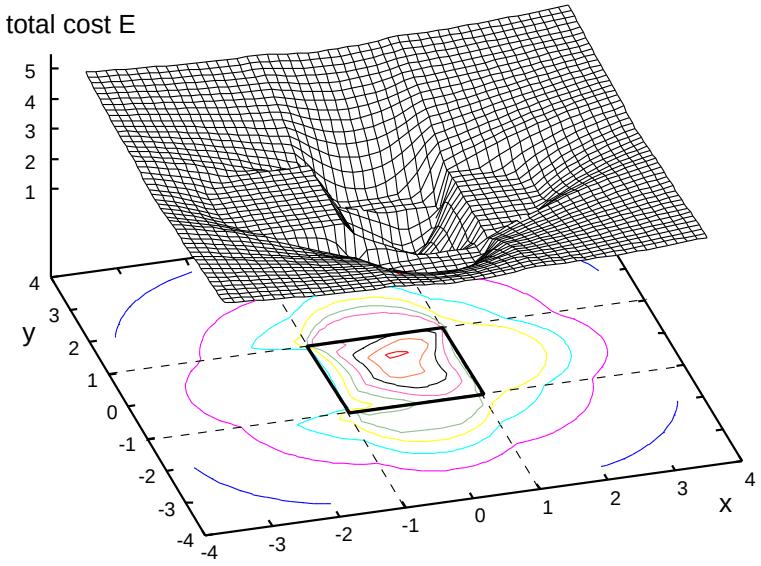
$$\begin{aligned} E =& |\beta_l - \alpha_l| + |\beta_t - \alpha_t| + |\beta_r - \alpha_r| + |\beta_b - \alpha_b| \\ =& |\beta_l - \alpha_l| + |\beta_t - \alpha_t| + |\beta_r - \alpha_r| + |\beta_l - \beta_t - \beta_r - \alpha_b|, \end{aligned} \tag{14.8}$$

we can resolve the absolute-value operators to derive

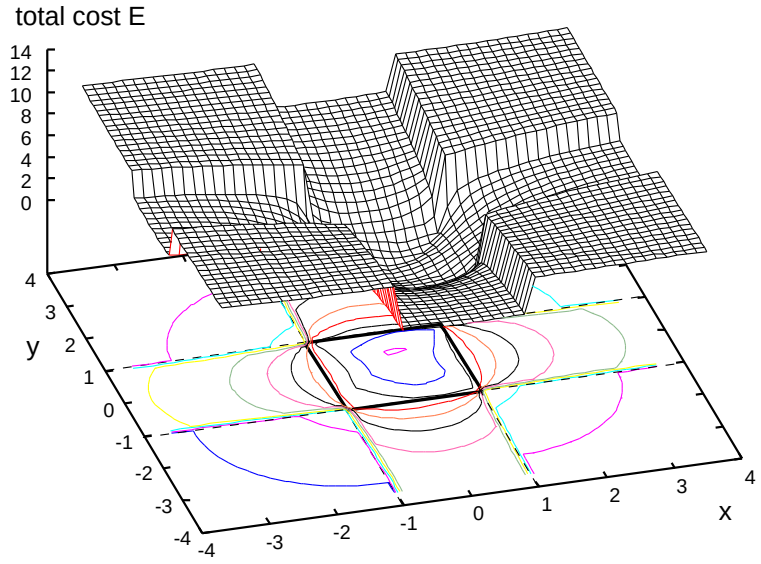
$$\begin{aligned} E =& (\beta_l - \alpha_l) - (\beta_t - \alpha_t) - (\beta_r - \alpha_r) - (\beta_l - \beta_t - \beta_r - \alpha_b) \\ =& -\alpha_l + \alpha_t + \alpha_r + \alpha_b, \end{aligned} \tag{14.9}$$

which is a constant. This explains the plateaus in the error function at each wall.

If we consider the same room geometry, but with the oriented angle, we obtain the error surface as depicted in Figure 14.17(b). This error function

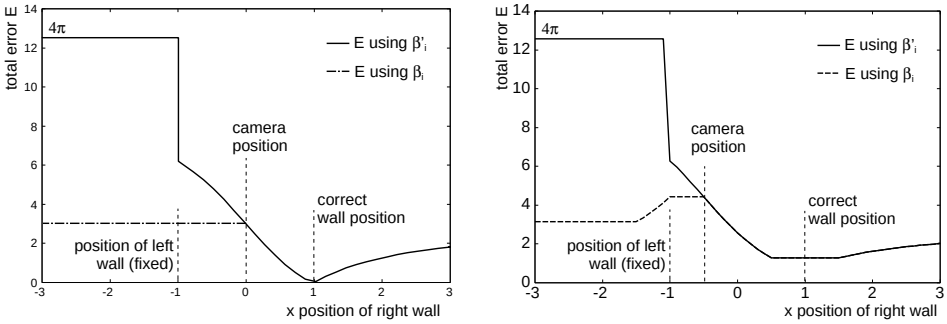


(a) Model error E computed with $\beta_{i,j,k}$.



(b) Model error E computed with $\beta'_{i,j,k}$.

Figure 14.17: *Model error E when moving the camera position while keeping the wall coordinates constant.*



(a) Error function with camera at correct position ($x = 0$).

(b) Error function with camera at incorrect position ($x = -0.5$).

Figure 14.18: *(a) Total model error for a rectangular room, depending on the position of the right wall. (b) If the order of the left and right wall is interchanged, it can lead to a constant error E.*

shows neither plateau areas nor local minima for varying camera positions. Instead, the error surface shows discontinuities whenever the camera crosses a wall plane, because in this moment, the oriented angle jumps between 0 and 2π. However, these steps in the error function impose no problem for the gradient descent search, since the step is always downwards in the direction to the minimum. Consequently, while the optimization can get stuck on the plateau areas using the inner angle definition, convergence is ensured with the oriented angle.

Dependency of the model error on the wall positions

A similar behaviour of the model error can be observed when keeping the camera position constant and varying the wall positions. We examine again the case of a rectangular room, with walls at $x, y = \pm 1$ and the real camera at $(0, 0)$. However, now we consider the position of the right wall as unknown and variable. Figure 14.18 depicts the resulting model error E that is obtained for each angle definition. In Figure 14.18(a), the camera is set to the correct position at $(0, 0)$, while it is set to $(-0.5, 0)$ in Figure 14.18(b). We can observe that the error function for the non-directed angles β show larger plateau regions and even local minima. Similar to the previous example, the directed angle definition β' leads to steps in the error function, but no local minima. Note that the left plateau area for oriented angles starts when the right wall position is moved so far to the left, that it crosses the left wall such that it is actually left of the left wall. For non-oriented angles, the plateau region already starts when the wall

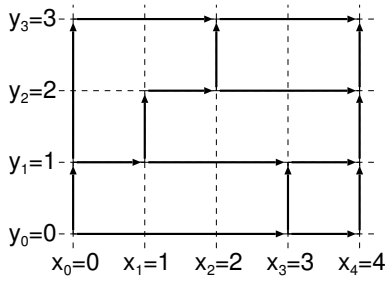


Figure 14.19: *An example initialization of a floor plan, based on the predefined ordering. Note that swapping x_2 and x_3 also gives a valid ordering.*

crosses the camera position.

We can conclude that the oriented angle β' shows clear advantages over the inner angle β. The oriented angle does not result in plateau regions for varying camera positions and it provides a clear error minimum. Both angle definitions lead to plateaus when the order of room walls is swapped, but for the oriented angles, these regions are smaller.

We conducted the same experiments with defining the error as the sum of squared angle differences. However, this definition leads to an error surface with many local minima, so that we did not pursue this further.

14.5.6 Initialization of the floor plan layout

In the last section, we observed that the error surface of E is smooth and has a unique minimum as long as the cameras are placed within the rooms, and as long as the order of the room walls are not interchanged. Hence, the optimization should be started with a configuration in which these conditions are satisfied to ensure convergence.

We obtain the initial placement of the walls by examining the user-specified floor plan model. Assuming that the walls are oriented along north-south or west-east direction, we can determine the direction going from one wall corner $\mathbf{p}_i$ to the other corner $\mathbf{p}_j$. Diagonal walls are not considered here. Based on this information, we build a west-east ordering of the points such that point $\mathbf{p}_i <_x \mathbf{p}_j$ if corner i is to the west of j. A similar ordering $<_y$ can be defined for the north-south direction. Subsequently, these orderings can be used to assign increasing coordinates. Note that the orderings do not necessarily impose a unique valid ascending enumeration of the coordinates. For example, in Fig. 14.19, the coordinates x_2 and x_3 could also be swapped and still fulfill $<_x$. Any of these admissible orderings