Rao-Blackwellized Importance Sampling of Camera Parameters from Simple User Input with Visibility Preprocessing in Line Space

Anonymous submission

Abstract

Users know what they see before where they are: it is more natural to talk about high level visibility information ("I see such object") than about one's location or orientation. In this paper we introduce a method to find in 3D worlds a density of viewpoints of camera locations from high level visibility constraints on objects in this world. Our method is based on Rao-Blackwellized importance sampling. For efficiency purposes, the proposal distribution used for sampling is extracted from a visibility preprocessing technique adapted from computer graphics.

We apply the method for finding in a 3D city model of Atlanta the virtual locations of real-world cameras and viewpoints.

1 Introduction

3D models of large real world outdoor and indoor locations become increasingly accessible. We are interested in quickly locating users or cameras locations inside of those models from high level visibility constraints, such as "I see" or "I do not see" object *x*. By providing immediately accessible visibility constraints users can be given back information regarding their location. This can also be used as an initial estimate for other optimizations.

For instance, one might want to find all apartment with a view on a particular monument. Google Earth has 3D models of several major US cities. Those models can be used to extract the apartments with the desired view. Users can also virtually see how the view looks like, and eventually go to the website of the real-estate agency which sells it. Other applications might be for robot localization. A robot might capture features of particular buildings if outdoor or walls if indoor. The robot could be located using a rough model of the world. Users might also want to locate photographs they took. They are just required to input the building they recognize. They might later discover images taken by other users at the same location. In the 4D-cities project, we are interested in generating 3D model from a large set of images. Localization of cameras given images provides initial estimates for further reconstruction methods.

Figure 1: A bimodal density in a city 3D model. The two modes are circled in red. The buildings required as visible are highlighted in color. Samples from this distribution are shown in Figure 6

Our approach is based on finding a density of camera locations: to a camera location is assigned its likelyhood. The density will be more localized in space as constraints are added. Figure 1 shows a bimodal density with two disjoints region of space satisfying constraints, although one in more likely than the other since samples have higher

weights. The density is inferred using Rao-Blackwellized importance sampling – importance sampling of improved quality by integrating out over the subspaces of orientations and focal length. In the case of large 3D models, we do not want to sample to whole space as only a small subset will satisfy the constraints. Bittner and Wonka in [1] precompute visibility of buildings on the ground plane for urban walkthroughs. The technique relies on solving the occlusion problem for regions of the ground plane (called "view cells" in the 2D case[1]) in the dual line space. By extending this technique to the spacial locations, we can use this preprocessed visibility as the base of the proposal distribution of our sampler.

A related area of work is the wide-baseline matching [2, 3, 4, 5, 6, 7] algorithms. They work best for images with similar viewpoints. However, such image might not be available in the case of large models, or with historical images as we are interested in the 4d-cities project. In the field of visibility, exact solution of the visibility problem is found using the Scale-Aspect graph[8, 9]. However this solution is computationally very expensive. Visibility has been extensively studied in Computer Graphics for accelerated rendering of 3D scene (for a review [10]). Techniques separates in 2 kinds: from-point and from-region. Preprocessing using the dual line space as in [1] alloinferedws efficient from-region preprocessing and can act as a good prior for the refinement at the point level that the sampling will constitute.

2 Density Estimation

A Bayesian density will be used to represent knowledge about camera locations and orientations given a measurement. Through this density we will extract the most likely camera locations. The density however does not have a parametric form. We will make inference on it using Rao-Blackwellized importance sampling.

2.1 Modeling Assumption

We represent knowledge about camera locations with the density $P(\theta|Z)$, where t, R, f is the 7 dimensional triple made of the three dimensional translation t, rotation R and focal length f of a given viewpoint or camera. Z is the measurement: the variable which contains the visibility constraints. $P(\theta|Z)$ tells how likely is the orientation θ given the visibility constraint Z. It can be rewritten using Bayes' law as $P(\theta|Z) = \frac{1}{P(Z)}P(Z|\theta)P(\theta)$, where $P(\theta)$ is the likelyhood of θ independently of any constraints (for instance, always zero under the ground plane).

2.2 Inference through Importance Sampling

This density $P(\theta|Z)$ does not have a parametric form and thus needs to be estimated. Importance sampling is a way of making approximate inference about a density $P(\theta|Z)$. First, samples $\theta_1, \theta_2, ..., \theta_r$ are generated from a *proposal distribution* $Q(\theta|Z)$. The proposal distribution is an initial hypothesis on the density $P(\theta|Z)$, given some knowledge. The approximate inference $\hat{P}(\theta|Z)$ on $P(\theta|Z)$ is obtained by weighting each sample θ_i by an importance weight $w_i = \frac{P(\theta_i|Z)}{Q(\theta_i|Z)}$:

$$\hat{P}(\theta|Z) = \sum_{i=1}^{r} w_i \delta(\theta_i, \theta)$$

where $\delta(\theta_i, \theta) = 1$ when $\theta = \theta_i$, and 0 if not. We compute the weights using the formula $P(\theta_i|Z) = \frac{1}{P(Z)}P(Z|\theta_i)P(\theta_i)$, by setting $P(Z|\theta_i)$ to 1 is the constraints Z is satisfied under the location, orientation and focal length $\theta_i = t_i, R_i, f_i$, and 0 if not.

2.3 Rao-Blackwellized Importance Sampling

The quality of the inference provided by the importance sampling can be improved using Rao-Blackwellization. We can rewrite the distribution as $P(\theta|Z) = P(R, f|Z, t)P(t|Z)$. It is equivalent, for extracting θ, to first extract t given Z and then R, f given Z and t. The density $P(R, f|Z, t)$ is easy to approximate when t is known. One can make inference on t by sampling $\hat{P}(\theta|Z)$: this will give the estimator $\hat{t}(\theta = t, R, f) = \sum_{i=1}^{r} t\delta(\theta_i, \theta = t, R, f)$.

The Rao-Blackwell says that a better estimator than $\hat{t}(\theta)$ is $\check{t}(s) = \mathbb{E}[\hat{t}(\theta)|S(\theta) = s, Z]$ where S is a sufficient statistic for t_{real}, the hidden "real" location behind the density. A sufficient statistic simply says that $P(\theta|S, Z)$ is independent of t_{real} so that the estimator

$\check{t}(s) = \mathbb{E}[\hat{t}(\theta)|S(\theta) = s, Z] = \sum \hat{t}(\theta)P(\theta|S(\theta) = s, Z)$ is properly defined. In our case, a trivial sufficient statistic is $S(\theta = t, R, f) = t$. We have that $\check{t}(s) = \check{t}(t) = \sum_{t_i \in T_S} t\delta(t, t_i)$, with T_S the set of unique t_i in the samples $\theta_1, ..., \theta_r$, and $\check{t}(t)$ is seen with probability $P(t|Z)$.

This corresponds to sampling the distribution $\check{P}(t|Z) = \sum_{t_i \in T_S} \alpha_i \delta(t_i, t)$, where α_i are the new weights $\alpha_i = P(t_i|Z)$. We can compute them by integrating over R and f:

$$\begin{aligned} \alpha_i &= \int P(t_i, R, f|Z)dR, f \\ &= \frac{P(t_i)}{P(Z)} \int P(Z|t_i, R, f) \cdot P(R, f|t_i)dR, f \end{aligned}$$

using Bayesè law.

We can now just sample this distribution of higher quality to find a location t, and sample $P(R, f|Z, t)$ to find R, f.

3 Visibility Preprocessing

However, in the case of large models only a small subset of space will actually satisfy the visibility constraints. It will be inefficient to sample those regions as most of the samples will have zero weight. We thus precompute visibility for 3D blocks of space, and integrate this information in the proposal distribution of the importance sampling. Bittner and Wonka [1] preprocess visibility for regions of the ground plane in order to make tractable the real-time rendering of walkthrough through large city 3D models. We adapt this method to 3D space to extract the regions in space which satisfy our visibility constraints.

3.1 3D View Cells

We use 3D view cells to carve out the space into distinct regions for which we will extract visibility information. A 3D view cell is represented by a box in space of height h and width and height a. The space is partioned by a set of view cells centered at (ka, la, mh) where $k, l, m \in \mathbb{Z}$ (Figure 2). The 2.5 hypothesis of a 3D model consists in saying that every point can be described by the equation $z = f(x, y)$, where x, y evolve in the ground plane. This hypothesis is satisfied by 3D models of cities, where z will represent the height of the object located at x, y. This hypothesis implies for the view cell that a object is visible from within the view cell is and only if it is visible from its top square (when $z = (m + 1/2)h$). To find the object visible *at least* from somewhere in the view cells, one may just look at those which are visible from each horizontal square centered at $(ka, la, (m + 1/2)h)$. In the case of view cells of reasonable size (smaller than the objects in order not to contain any of them), we can also make the following assumption: a object visible from within the view cell top square will be also visible from the view cell top square *edges*. We can thus restrict the study of visibility to the study of all light rays reaching this set of edges (Figure 2).

3.2 Visibility in Line Space

This problem of finding visibility information for all light rays getting into a view cell is more easily solved in the dual line space.

First, we ignore the height information and consider the 2D plane containing a view cell top square. This plane contains a set of object edges – intersection of object faces with the plane. We would like to find all the lines which lies in this plane and intersect both the cell edge and the object edge. A line (a, b, c) in canonical space is associated to the point $(a/c, b/c)$ in line space. The set of all lines $l = a_l x + b_l y + 1$ passing through a view cell edge and a object edge (represented as the shaded region in figure 2) can be expressed as a barycenter $l = \sum_{l' \in AC,AD,BC,BD} \alpha_{l'}(a_{l'} x + b_{l'} y + 1)$ where $\sum_{l'} \alpha_l = 1$. In the dual line space, this means that all such lines are represented by the points contained within the polygon formed by the dual points of the 4 limit lines (Figure 2). A object edge e_1 will occlude another object edge e_2 for all points on a view cell edge c if and only if e_2 is located behind e_1 *and* if the polygon associated with the four critical lines of e_2 with c in line space is fully contained within the polygon of e_1. This means that all lines which are intersecting e_2 and the cell edge c are intersecting e_1 first.

However, this can only be generalized for visibility regarding if he objects are of same height, which is not the case in general. To integrate the height information, we compute the maximum angle with the ground plane made by the cell edge extremities (at the height $(m + 1/2)h$))

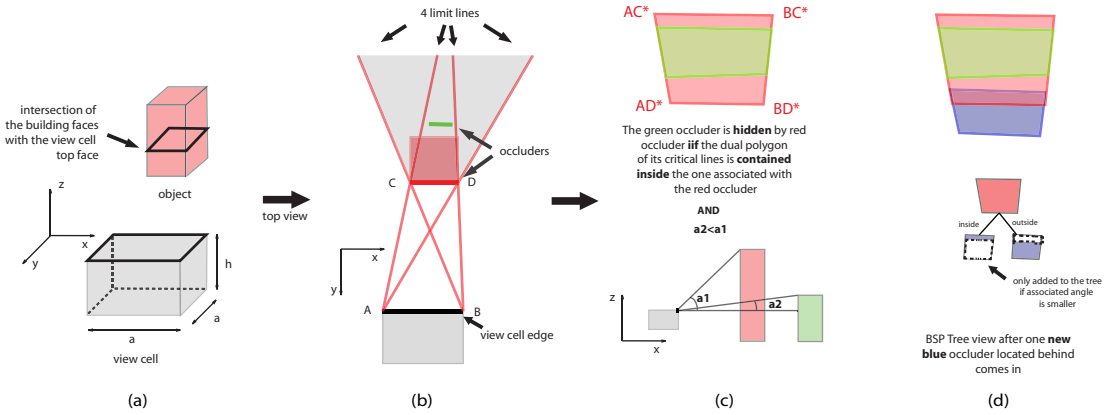

Figure 2: (a) A 3D view cell and a 3D object. The intersection of the object faces with the plane containing the view cell top edge creates occluders on this plane. (b) Lmit lines between a view cell edge and an occluder. All lines in the shaded region are "average" of those lines. Behind is one hidden green occluder, for which no limit lines are shown. (c) Line space representation of the 4 limit lines of the red occluder and green occluder. The polygon of the green occluder will be contained inside the one of the red occluder, as it is hidden by the red occluder. (d) A schematic representation of the BSP tree when a new blue occluder comes in.

and all of the object edge extremities at the height of the object. All locations of the view cell edge will see the object under a maximum angle below that computed angle. A face of a object f_1 is occluded by another face f_2 for a all points on a view cell c is the projection of f_1 on the view cell plane is occluded by the projection of f_2 *and* if the maximum angle with the ground plane of f_1 is inferior to the maximum angle of f_2 (Figure 2). Figure 3 shows the occlusion mechanism in line space.

3.3 Iterative Solving of Visibility with BSP Trees

We use Binary Space Partitioning trees (BSP trees) to test in line space whether a polygon is hidden by a set of others and to maintain the occluders structure. BSP are convenient for making inside/outside comparisons. For simplification the tree can be seen as if each node correspond to a polygon (in fact, they are the lines generating the polygons). Each node have a "left" descendant corresponding to the tree containing all the nodes located inside, and a "right" descendant corresponding to the tree containing all the nodes located outside. When a new polygon

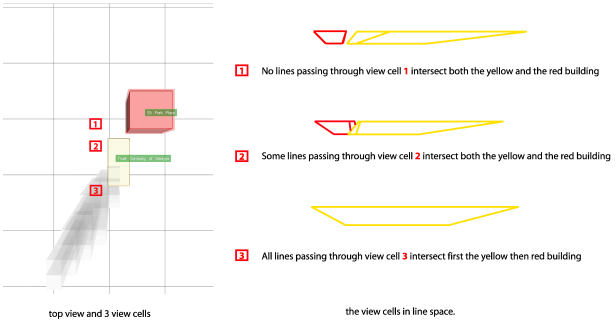

Figure 3: Seeing occlusion in line space

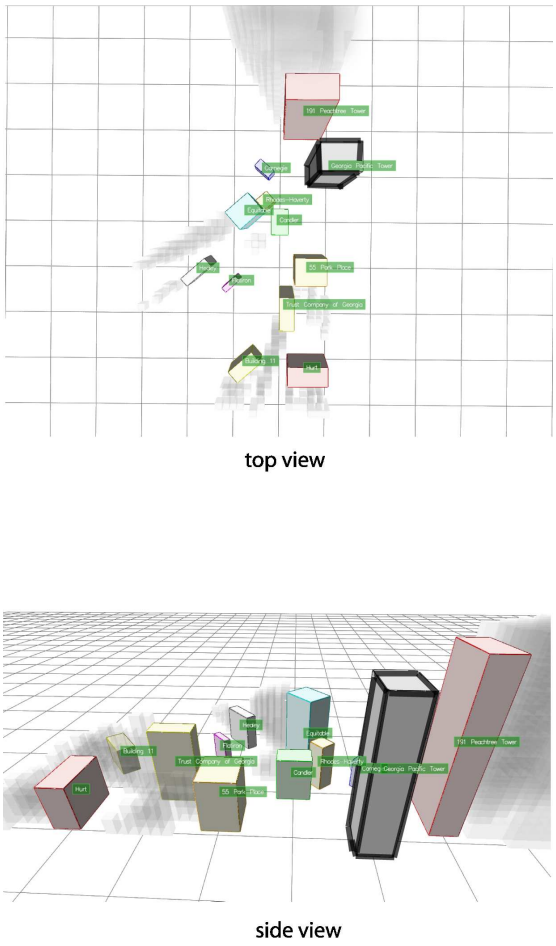

Figure 4: Rough 3D model of Atlanta used for testing. Shaded regions corresponds to view cells which does not see the building highlighted in black.

comes, if it lies inside, it is added to the left node, if it lies outside it is added to the right node, and if it has outside and inside parts, it is split in two inside and outside parts added to the left and right descendant respectively (Figure 2). We prune at construction the "inside" nodes if their associated angle (for the object's height) is smaller than those of their father, and add the polygons in front to back order. If an object is visible, it will be remain in the tree.

Given a new visibility constraint, one can now simply find which 3D view cells might satisfy the constraint by exploring the tree. Figure 4 shows the result of such preprocessing.

4 Experiments

We tested our method using a rough 3D model of downtown Atlanta, containing 12 landmark buildings. We only looked at ground plane and building locations (Figure 5). This is equivalent to rewriting the prior on θ as the mixture $P(\theta) = \alpha P_{ground}(\theta) + \beta P_{buiding}(\theta)$, where $P_{ground}(\theta)$ and $P_{building}(\theta)$ denotes the prior respectively of θ for ground locations and for locations over the buildings faces. Information about orientation is integrated in the priors: for instance, only "up" orientations are valid for the ground plane, and only "outside" orientations are valid for buildings. The parameter α and β are adjusted regarding whether we are looking for ground or building solutions.

Figure 5 presents results when the system only has a 2 constraints, for instance solving the problem "I want to test the views seeing those two particular buildings". The distribution is mutimodal as more than one region of space satisfy such constraint. Such density could be weighted by other information, for instance in order to only sample to the locations corresponding to an apartment that one is trying to sell.

As constraints are added, we observe that the density goes from multimodal to unimodal (Figure 6). Figure 6 shows localization of photographs taken in downtown Atlanta. Sampling the density extracted for each of those photographs gives often views close to the actual one. This confirms our modeling assumptions.

Results from Figure 7 shows localization of images taken from the web (flickr.com). Interestingly, the first

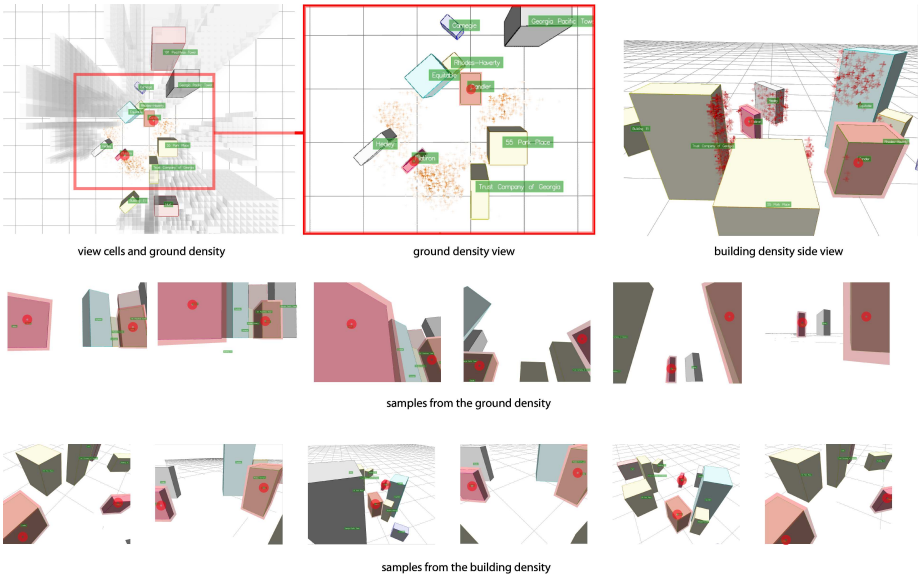

Figure 5: Exploring views which satisfy visibility constraints. Red dots correspond to buildings required as visible. First samples line correspond to ground location, second to building locations.

image is manually tagged on flickr with most of the building names. We used those tags to find a probable location of the person taking the picture. Samples are slightly biased towards the left as the two rightmost buildings are not tagged. The second image correspond to another image extracted from flickr that we manually tagged. Again, we find the building from which this picture has been taken form (191 Peachtree Tower).

5 Discussion

We introduced a method to efficiently estimate camera positions or view points from simple visibility constraints. Thanks to visibility preprocessing in line space, the method is scalable to large 3D models. We demonstrated its accuracy for localizing photographs in a city model with as little as 12 buildings. The method will keeps on getting more accurate as new objects are added in models, as they will restrict even more the regions satisfying users visibility constraints.

References

[1] Bittner, P. Wonka, and M. Wimmer, "Visibility preprocessing for urban scenes using line space subdivision," in *Proceedings of Pacific Graphics 2001 (Ninth Pacific Conference on Computer Graphics and Applications)* (B. Werner, ed.), (Los Alamitos, CA), pp. 276–284, IEEE Computer Society Press, Oct. 2001.

[2] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 5, pp. 530–535, 1997.

[3] P. Pritchett and A. Zisserman, "Wide baseline stereo matching," in *Intl. Conf. on Computer Vision (ICCV)*, pp. 754–760, 1998.

[4] P. Pritchett and A. Zisserman, "Matching and reconstruction from widely separated views," in *SMILE 98 European Workshop on 3D Structure from Multiple Images of Large-Scale Environments, Freiburg, Germany*, 1998.

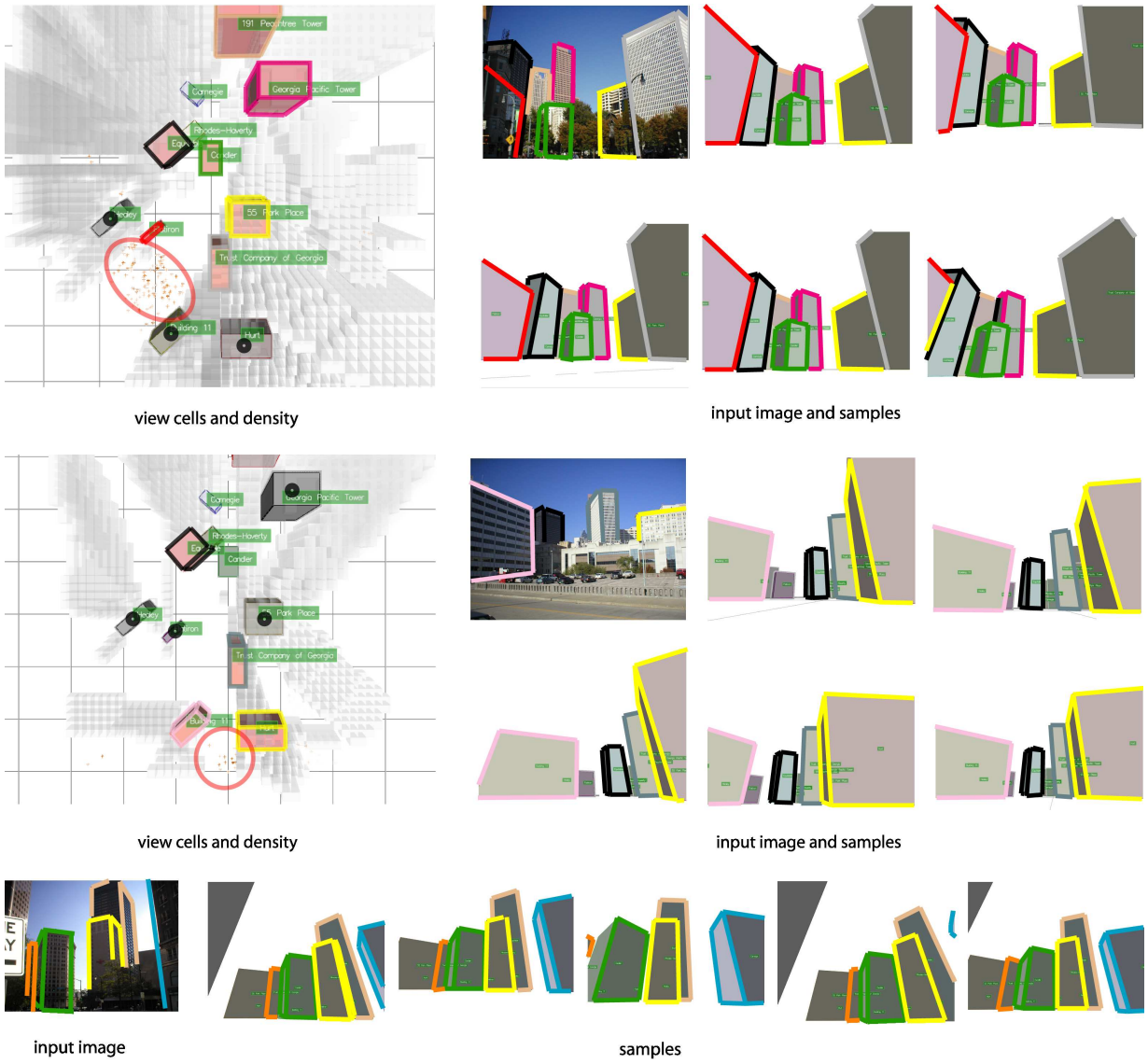

Figure 6: Localizing photographs. We highlighted in color on both the photograph and the samples the buildings set as visible. On the map, the building with black dots are the one set as not visible. The last line corresponds to samples from the density of Figure 1

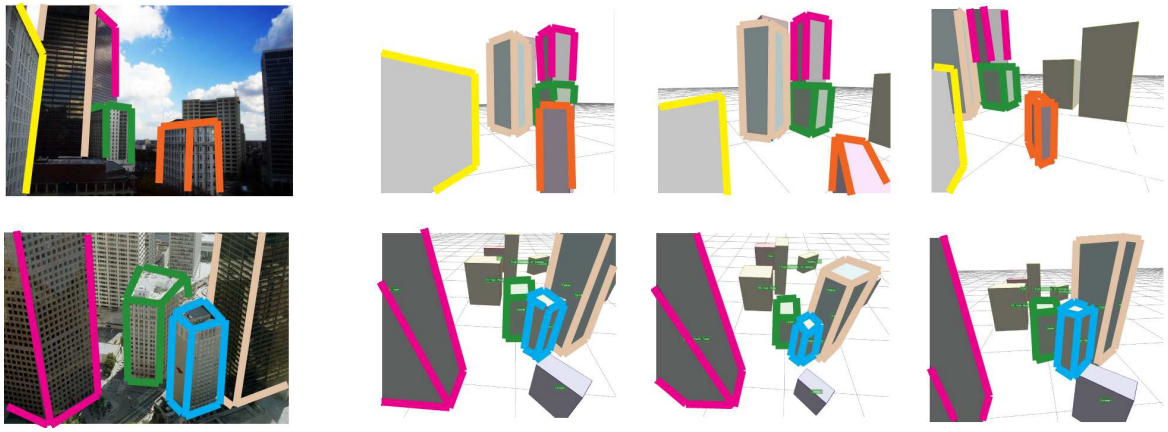

Figure 7: Localizing photographs. Images are from flickr.com. We used the tags (building names) on the image on flick to locate the first images. We manually tagged the second one. We highlighted in color on both the photograph and the samples the buildings set as visible.

[5] A. Baumberg, "Reliable feature matching across widely separated views," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 774–781, 2000.

[6] T. Tuytelaars and L. Van Gool, "Matching widely separated views based on affinely invariant neighbourhoods," in *British Machine Vision Conference (BMVC)*, pp. 412–422, 2000.

[7] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *British Machine Vision Conference (BMVC)*, pp. 414–431, 2002.

[8] H. Plantinga and C. R. Dyer, "Visibility, occlusion, and the aspect graph," *Int. J. Comput. Vision*, vol. 5, no. 2, pp. 137–160, 1990.

[9] D. W. Eggert, K. W. Bowyer, C. R. Dyer, H. I. Christensen, and D. B. Goldgof, "The scale space aspect graph," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 11, pp. 1114–1130, 1993.

[10] D. Cohen-Or, Y. Chrysanthou, and C. Silva, "A survey of visibility for walkthrough applications," 2000.