Inferring Temporal Order of Images From 3D Structure

Grant Schindler Frank Dellaert
Georgia Institute of Technology
{schindler,dellaert}@cc.gatech.edu

Sing Bing Kang
Microsoft Research, Redmond, WA
SingBing.Kang@microsoft.com

Abstract

In this paper, we describe a technique to temporally sort a collection of photos that span many years. By reasoning about persistence of visible structures, we show how this sorting task can be formulated as a constraint satisfaction problem (CSP). Casting this problem as a CSP allows us to efficiently find a suitable ordering of the images despite the large size of the solution space (factorial in the number of images) and the presence of occlusions. We present experimental results for photographs of a city acquired over a one hundred year period.

1. Introduction

Cameras and skyscrapers have now coexisted for more than a century, allowing us to observe the development of cities over time. We are interested in being able to automatically construct a time-varying 3D model of a city from a large collection of historical images. Such a model would reflect the changing skyline of the city, with buildings created, modified, and destroyed over time. It would also be useful to historians and urban planners both in organizing collections of thousands of images (spatially and temporally) and in generating novel views of historical scenes by interacting with the time-varying model itself.

To extract time-varying 3D models of cities from historical images, we must perform inference about the position of cameras and scene structure in both space and time. Traditional structure from motion (SFM) techniques can be used to deal with the spatial problem, while here we focus on the problem of inferring the temporal ordering for the images as well as a range of dates for which each structural element in the scene persists. We formulate this task as a constraint satisfaction problem (CSP) based on the visibility of structural elements in each image. By treating this problem as a CSP, we can efficiently find a suitable ordering of the images despite the large size of the solution space (factorial in the number of images) and the presence of occlusions.

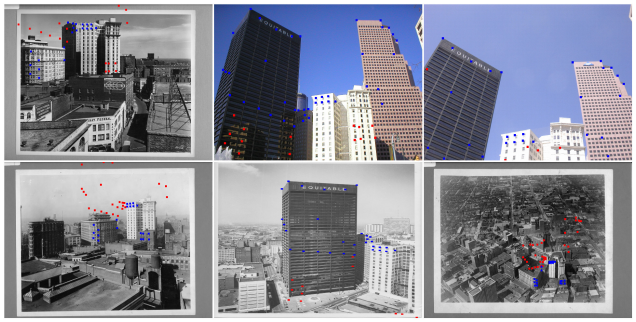


Figure 1. Given an unordered collection of photographs, we infer the temporal ordering of the images by reasoning about the visibility of 3D structure in each image.

2. Related Work

SFM is now a well-studied problem, and the early stages of our approach proceed very much in the same manner as in [11], recovering calibrated cameras and the 3D point locations based on 2D correspondences between images. Time-varying SFM problems have been studied in the context of ordered image-sequences of objects in motion [7], while we work with an unordered (both spatially and temporally) collection of images. Although reasoning about visibility and occlusions has previously been applied to view synthesis from multiple images [8], surface reconstruction [13], and model-based self-occlusion for tracking [10], it has not been used in the context of temporal sorting.

The earliest work on temporal reasoning involved the development of an interval algebra describing the possible relationships between intervals of time [1]. A number of specific temporal reasoning schemes were later captured by temporal constraint networks [3] which pose the temporal inference problem as a general constraint satisfaction problem. Such networks are often used for task scheduling, given constraints on the duration and ordering of the tasks. Efficient solutions to temporal constraint networks rely on sparsity in the network, whereas our problem amounts to handling a fully connected network. Uncertainty was later introduced into temporal constraint networks [4, 5, 2] by relaxing the requirement that all constraints be fully satisfied.

1-4244-1180-7/07/$25.00 ©2007 IEEE

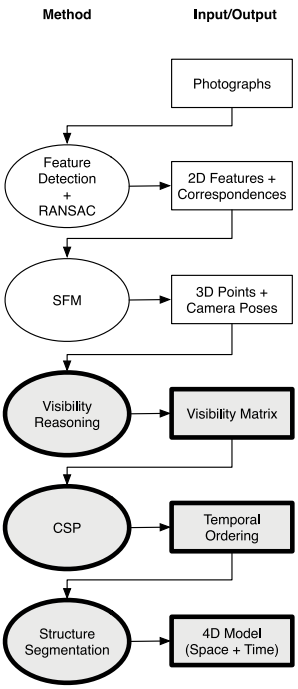


Figure 2. Overview of Approach. A fully automated system for building a 4D model (3D + time) of a city from historical photographs would consist of all these steps. Here, we concentrate on the highlighted steps of visibility reasoning and constraint satisfaction to infer a temporal ordering of images which can then be used to construct the 4D model.

3. Overview of Approach

We are interested in inferring the temporal ordering of images as one step in a system for producing time-varying 3D models of cities from historical photographs. As summarized in Figure 2 the process begins by performing feature detection and matching on a set of input photographs, followed by SFM to recover 3D points and camera poses. The feature detection and SFM steps are beyond the scope of this paper and we do not discuss them in detail here, other than to say that in this work the feature detection and matching are performed manually, while SFM is automatic.

In this paper, we focus on the problems of visibility reasoning, temporal ordering, and time-varying 3D model construction as highlighted in Figure 2. Our method takes 3D points and camera poses as input and uses them to compute a matrix describing the visibility of each 3D point in each image (Section 4). The temporal ordering of the images is then recovered by reordering the columns of this visibility matrix in a CSP framework (Section 5). Finally, the inferred temporal ordering is used to visualize a 4D model (space + time) of the changing city (Section 6).

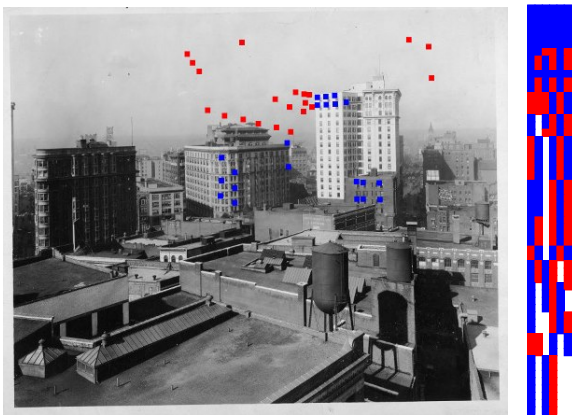


Figure 3. Point Classification. In each image, every 3D point is classified as $observed$ (blue), $missing$ (red), $out\,of\,view$ (white) or $occluded$ (white). The missing points belong to buildings that do not yet exist at the time the photograph was taken. Classifications across all images are assembled into a visibility matrix (right) which is used to infer temporal ordering. Each column of the visibility matrix represents a different image, while each row represents the visibility of a single 3D point across all images.

4. Visibility Reasoning

The problem we will address is inferring the temporal ordering of a set of n un-ordered images $I_{1..n}$ registered to a set of m 3D points $X_{1..m}$. The key to inferring temporal order from a collection of historical urban images is that different sets of 3D structures exist in the same place in the world at different points in time. Thus, we must determine which structures exist in each image, and to do this we must reason about the visibility of each 3D point in each image. We show here how to encode the information provided by each image I_j about every 3D point X_i in a visibility matrix.

4.1. Visibility Classification

To determine whether a building exists at the time an image was taken, we reason about the visibility of each 3D point on that building. Assuming known projection matrices $P_{1..n}$ for each of the n cameras $C_{1..n}$ corresponding to images $I_{1..n}$, every 3D point can be classified in each image as $observed$, $missing$, $out\,of\,view$, or $occluded$ as follows. If a measurement u_{ij} exists for point X_i in image I_j, the point is $observed$. If the projection $x_{ij} = P_j \begin{bmatrix} X_i \\ 1 \end{bmatrix}$ of point X_i in image I_j falls outside the field of view of the camera (as defined by the width and height of the corresponding image), the corresponding point is classified as $out\,of\,view$ for that image. If the projection x_{ij} is within the field of view of the camera but no measurement u_{ij} exists, the point may be classified either as $missing$ or $occluded$, and further work is required to determine which classification is correct (see Section 4.2).

The intuition behind this classification is that we want to know whether the physical structure corresponding to point X_i existed at the time that image $I_j$was captured. If it does not appear where we expect it to be, either it did not exist at the time ($missing$) or else something is blocking our view of it ($occluded$). We discuss how to distinguish between these two cases in the next section.

4.2. Occlusion

We can also use occlusion reasoning to determine why a building might not appear in a given image. To this end, we assume that the 3D points $X_{1..m}$ correspond to a sparse sampling of the surface of a number of solid structures in the scene. For every triplet of points, the triangle $X_a X_b X_c$ that they define may or may not lie along the surface of a solid structure. If we can find a triangulation of these points that approximates the solid structure, the faces of such a mesh will occlude the same set of points occluded by the physical structure, and these occluding faces can be used to distinguish between points that are $missing$ and $out\ of\ view$.

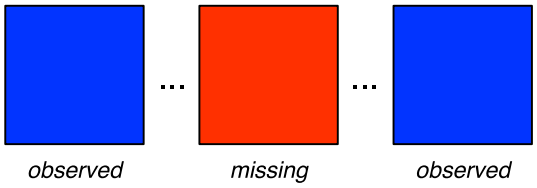
Inspired by [6] and the image-consistent triangulation method of [9], we proceed as follows: For each image I_j, we compute the Delaunay triangulation of the measurements u_{ij} in that image. Each 3D triangle corresponding to a face in the Delaunay triangulation is a potential occluder and for each triangle, we test whether it fails to occlude any $observed$ points in the scene. That is, if a face is intersected by a line segment $O_j X_i$ from any camera's center of projection O_j to any observed 3D point X_i corresponding to a measurement u_{ij}, it is removed from the potential pool of occluders. The intuition behind this approach is that if the triangle was a true occluder, it would have blocked such a measurement from being observed. After testing all faces against all observed points X_i in all images I_j, we are left with a subset of triangles which have never failed to block any 3D point from view, and we treat these as our occluders.

To determine whether a point X_i is $missing$ or $occluded$ in a given image I_j, we construct a line segment from the center of projection O_j of camera C_j to the 3D point X_i. If this line segment $O_j X_i$ intersects any of the occluding triangles, the point is classified as $occluded$. Otherwise the point is classified as $missing$, indicating that the point X_i did not exist at the time image $I_j$was captured.

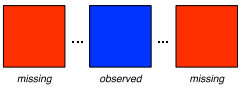
4.3. Visibility Matrix

Finally, we can capture all this information in a convenient data structure—the visibility matrix. We construct an $m \times n$ visibility matrix V indicating the visibility of point X_i in image I_j as

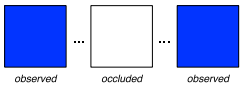
$$v_{ij} = \begin{cases} +1 & if\ X_i\ is\ observed\ in\ I_j \\ -1 & if\ X_i\ is\ missing\ in\ I_j \\ 0 & if\ X_i\ is\ out\ of\ view\ or\ occluded\ in\ I_j \end{cases}$$



(a) violates constraints



(b) satisfies constraints



(c) satisfies constraints

Figure 4. Visibility constraints. The columns of the visibility matrix must be reordered such that the situation in (a) never occurs – it should never be the case that some structure is visible, then vanishes, then appears again. Rather, we expect that buildings are constructed and exist for some amount of time before being demolished as in (b). Note that the constraint in (a) does not rule out the situation in (c) where structure becomes occluded.

See Figure 3 for an example of such a visibility matrix. In all figures, the value $+1$ is indicated with a blue dot, -1 with a red dot, and 0 with a white dot. Note that the columns of such a matrix correspond to entire images, while the rows correspond to single 3D points.

5. Constraint Satisfaction Problem

We pose the temporal ordering problem as a constraint satisfaction problem (CSP), where constraints are applied to the visibility matrix of the given scene. Specifically, once a visibility matrix V is constructed, the temporal ordering task is transformed into the problem of rearranging the columns of V such that the visibility pattern of each point is consistent with our knowledge about how buildings are constructed. Our model assumes that every point X_i is associated with a building in the physical world, and that buildings are built at some point in time T_A, exist for a finite amount of time, and may be demolished at time T_B to make way for other buildings. We also assume that buildings are never demolished and then replaced with an identical structure. These assumptions gives rise to constraints on the patterns of values permitted on each row in V.

The constraints on the visibility matrix can be formalized as follows: on any given row of V, a value of -1 may not occur between any two $+1$ values. This corresponds to the expectation that we will never see a building appear, then disappear, then reappear again, unless due to occlusion or being outside the field of view of the camera (see Figure 4). The valid image orderings are then all those that do not violate this single constraint.

Because we have expressed the temporal ordering prob-

Figure 5. Local Search starts from a random ordering and swaps columns and groups of columns in order to incrementally decrease the number of constraints violated. Here, 30 images are ordered by taking only 10 local steps.

lem in terms of constraints on the visibility matrix, we can use the general machinery of CSPs to find a solution. A common approach to CSPs is to use a recursive backtracking procedure which explores solutions in a depth first search order by assigning an image I_j to position 1, then another image to position 2, etc. At each step, the partial solution is checked and if any constraints are violated, the current branch of search is pruned and the method "backtracks" up one level to continue the search, having just eliminated a large chunk of the search space. Given that our problem has $n!$ solutions (i.e., factorial in the number of images n), this method becomes computationally intractable for even relatively small numbers of images.

5.1. Local Search

CSPs can also be solved using a local search method to get closer and closer to the solution by starting at a random configuration and making small moves, always reducing the number of constraints violated along the way. This solution has been famously applied to solve the n-queens problem for 3 million queens in less than 60 seconds [12].

For our problem, a local search is initialized with a random ordering of the images, corresponding to a random ordering of the columns in the visibility matrix V. At each step of the search, all local moves are evaluated. In our case, these local moves amount to swapping the position of two images or of two groups of images by rearranging the columns of the matrix V accordingly. In practice, swapping larger groups of images allows solutions to be found more quickly, preserving the progress of the search by keeping constraint-satisfying sub-sequences of images together.

During local search, we consider a number of candidate orderings of the columns of the visibility matrix, where different arrangements of columns will violate different numbers of constraints. As described above, a constraint is violated if, on a given row, a point is classified as $missing$ between two columns in which it was $observed$. The best local move is then the move that results in the ordering that violates the fewest constraints of all the candidate local moves being considered. If there is no move which decreases the number of constraints violated, we reinitialize the search with a random ordering and iterate until a solution is found. Once an ordering of the columns is found that violates no constraints, the temporal ordering of the images is exactly the ordering of the columns of the visibility matrix. Figure 5 demonstrates the progress of such a local search.

5.2. Properties of Ordering Solutions

Solving the above constraint satisfaction problem may give us more than just one possible temporal ordering of the images. For the n images, there may be r eras in which different combinations of structures coexist. If $r < n$, there is more than one solution to the constraint satisfaction problem. In particular, any two images captured during the same era may be swapped in the ordering without inducing any constraint violations in the visibility matrix.

In addition, there is a second class of solutions for which time is reversed. This is because any ordering of the columns that satisfies all constraints will still satisfy all constraints if the order of the columns is reversed. In practice, one can ensure that time flows in the same direction for all solutions by arbitrarily specifying an image that should always appear in the first half of the ordering. This is analogous to the common technique of fixing a camera at the origin during structure from motion estimation.

5.3. Dealing with Uncertainty

The above formulation depends upon an explicit decision as to the visibility status of each point in each image, and cannot deal with misclassified points in the visibility matrix. For example, if a point is not observed in an image, it is crucial that the point receives the correct label indicating whether the point no longer existed at the time the image was taken, or whether it was simply occluded by another building. If a single point is misclassified in one image, it may cause all possible orderings to violate at least one constraint, and the search will never return a result.

The ideal case, in which there are no occlusions and no points are out of view, will rarely occur in practice and there are a number of ways a point might be misclassified:

- Points that really should have been $observed$ might, due to failure at any point during automated feature detection or due to missing or damaged regions of historical images, be classified as $missing$.
- Points that were $occluded$ by un-modeled objects (such as trees or fog) may falsely be labeled $missing$.
- Points that were really $occluded$ may fail to be blocked by occlusion geometry due to errors in SFM estimation, and instead be falsely labeled as $missing$.
- Points that are truly $missing$ may be falsely explained away as $occluded$.

In practice, some combination of all these errors may occur.

We achieve robustness to misclassified points without introducing any additional machinery. CSPs can implicitly cope with this kind of uncertainty by relaxing the requirement that all constraints be satisfied. We modify the local search algorithm to return the ordering that satisfies more constraints than any other after a fixed amount of searching. Under such an approach, we can no longer be absolutely certain that the returned solution is valid, but we gain the ability to apply the approach to real-world situations.

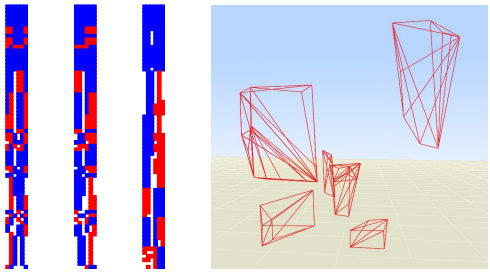
5.4. Structure Segmentation

In order to build a convincing 3D model of a city, we need to segment the 3D point cloud that results from SFM into a set of solid structures. In fact, we can use the visibility matrix V to extract such a building segmentation directly from the recovered image ordering. Once the columns have been reordered using local search, similar visibility patterns become apparent across the rows of the matrix. This is due to the fact that multiple 3D points originate from the same physical structures in the world, and thus come in and out of existence at the same time. This is made more apparent by reordering the rows of the visibility matrix to group points that share times of appearance T_A and disappearance T_B. Such a reordering amounts to segmenting the 3D point cloud into disjoint sets. By taking the 3D convex hull of each cluster of points, we get approximate scene geometry which can be textured and used for further synthesis of new views in space and time (see Figure 6).

6. Results

We tested our method on a set of images of a city collected over the period from 1897 to 2006. For the results presented here, feature detection and matching were performed manually. Given a set of 2D correspondences across images, the remaining steps of the algorithm beginning with SFM (see Figure 2) are performed automatically.

In our first experiment, we find a temporal ordering for 6 images of a scene containing 56 3D points (Figure 7). In this case, we purposely chose photographs with clear views of all structure points, meaning that none of the points are misclassified in the visibility matrix and an exact solution to the ordering is guaranteed. Due to the small number of images, we perform an exhaustive back-tracking search to find all possible ordering solutions. Back-tracking search finds that out of the $6! = 720$ possible orderings, there are 24 orderings which satisfy all constraints, one of which is shown in Figure 7. The 24 solutions are all small variations of the same ordering—images 1 and 2 may be interchanged, as may images 4, 5, and 6, and finally the entire sequence may be reversed such that time runs backwards. For this small problem, the search takes less than one second.



(a) (b) (c) (d)

Figure 6. Structure Segmentation. Beginning from a random ordering of the visibility matrix (a), local search re-orders the columns to the correct temporal ordering (b), and then rows are re-ordered to group 3D points that appear and disappear at the same times (c). We compute 3D convex hulls of each group of points to get solid geometrical representations of buildings in the scene (d).

In our second experiment, we deal with a more difficult group of 20 images of a scene consisting of 92 3D points (Figure 8). These images contain a number of misclassified points due to occlusions by trees and un-modeled buildings, as well as errors in the estimation of 3D point locations and camera positions by SFM. As such, we do not expect to find an ordering that satisfies all constraints, so we instead use 1000 iterations of local search to find the ordering which violates the fewest constraints. For each iteration of local search, we begin from a new random ordering of the images. Note the number of iterations of search (1000) is considerably smaller than the number of possible orderings, in this case $20! \approx 2.4 \times 10^{18}$. This local search returns an ordering (Figure 8) for which constraints are violated on 15 of the 92 rows of the visibility matrix. In the absence of any ground truth dates for the images, and with no exact solution to the CSP in this case, it can be difficult to evaluate the quality of the returned ordering. However, despite the large number of constraints violated, the ordering returned is consistent both with the sets of buildings which appear in each image and with the known dates of construction and demolition for all modeled buildings in the scene. The ordered visibility matrix for this experiment is shown in Figure 9.

In our third experiment, to simulate a larger problem, we synthesize a scene containing 484 randomly distributed 3D points and 30 cameras placed in a circle around the points. Each point is assigned a random date of appearance and disappearance, while each camera is assigned a single random date at which it captures an image of the scene. The resulting synthetic images only show the 3D points that existed on the date assigned to the corresponding camera. The size of the solution space ($30! = 2.65 \times 10^{32}$) necessitates local search for this problem. Starting from a random ordering, a solution that violates no constraints is found just 26 local moves away from the random initialization, taking less than one minute of computation. In contrast to the previous

Figure 7. Inferred temporal ordering of 6 images. In the case where there are no occlusions of observed points, we can guarantee that a solution exists that violates no constraints. The ordering shown is one of 24 orderings that satisfy all constraints. The other solutions involve swapping sets of images that depict the same set of structures and reversing the direction of time.

Figure 8. Inferred ordering of 20 images. Despite many misclassified points, the presence of un-modeled occlusions such as trees, and a solution space factorial in the number of images ($20! \approx 2.4 \times 10^{18}$), an ordering consistent with the sets of visible buildings is found by using local search to find the ordering that violates the fewest constraints. In such a case, there is no single solution which satisfies all constraints simultaneously.

experiment, a solution is quickly found for this synthetic scene (without the need to reinitialize the search) because no points are misclassified for the synthesized images.

Finally, we use the structure segmentation technique described in Section 5.4 to automatically create a time-varying 3D model from the 6 images in Figure 7. After ordering the columns of the visibility matrix to determine temporal order, we reorder the rows to group points with the same dates of appearance and disappearance. We then compute the convex hulls of these points and automatically texture the resulting geometry to visualize the entire scene (see Figure 10). Textures are computed by projecting the triangles of the geometry into each of the 6 images and warping the corresponding image regions back onto the 3D geometry.

7. Discussion

The computation time required for local search depends upon several factors. The main computational cost is computing the number of constraints violated by a given ordering of the visibility matrix, which increases linearly with m the number of points in the scene and n the number of images being ordered. In addition, at each step of local search, the number of tested orderings increases with n^2 since there are $\frac{(n)(n-1)}{2}$ ways to select two images to be swapped.

As demonstrated in the above experiments, the amount of computation also varies inversely with the number of valid orderings for a given visibility matrix. For ordering problems that admit many solutions, the random initialization of local search will often be close to some valid ordering, and will thus solve the problem quickly. This is, in fact, the key to the success of local search on the n-queens problem of [12], where the number of solutions actually increases with the size of the board. However, when there are very few solutions (or no exact solution, as in the above 20-image experiment), local search may require a large number of iterations until a random ordering is chosen that can reach the true solution using only local moves.

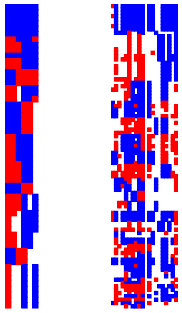


Figure 9. Ordered visibility matrices for sets of 6 images (left) and 20 images (right). The ordering of the 6 images on the left was found with backtracking search and satisfies all constraints. The ordering of the 20 images on the right violates the fewest constraints of all solutions found with 1000 iterations of local search. In the latter case, misclassified points caused by un-modeled occlusions lead to a situation in which no ordering can simultaneously satisfy all constraints.

Finally, note that the nature of the dates we infer for scene structure is abstract. For example, consider the building depicted in the first image in Figure 8. Rather than inferring that this building existed from 1902 to 1966, we can only infer that it existed from the time of Image 1 to the time of Image 13 (where images are numbered by their position in the inferred temporal ordering). Without additional knowledge, this is the most we can confidently say about when the building existed. When a human inspects a historical photograph, he or she may assign a time to it by identifying objects in the scene with known dates of existence—this may include known buildings, but also more abstract concepts such as the style of automobiles, signs, or the clothing of people depicted in the image. This suggests that a machine learning approach may be required if we hope to assign estimates of absolute dates to each image.

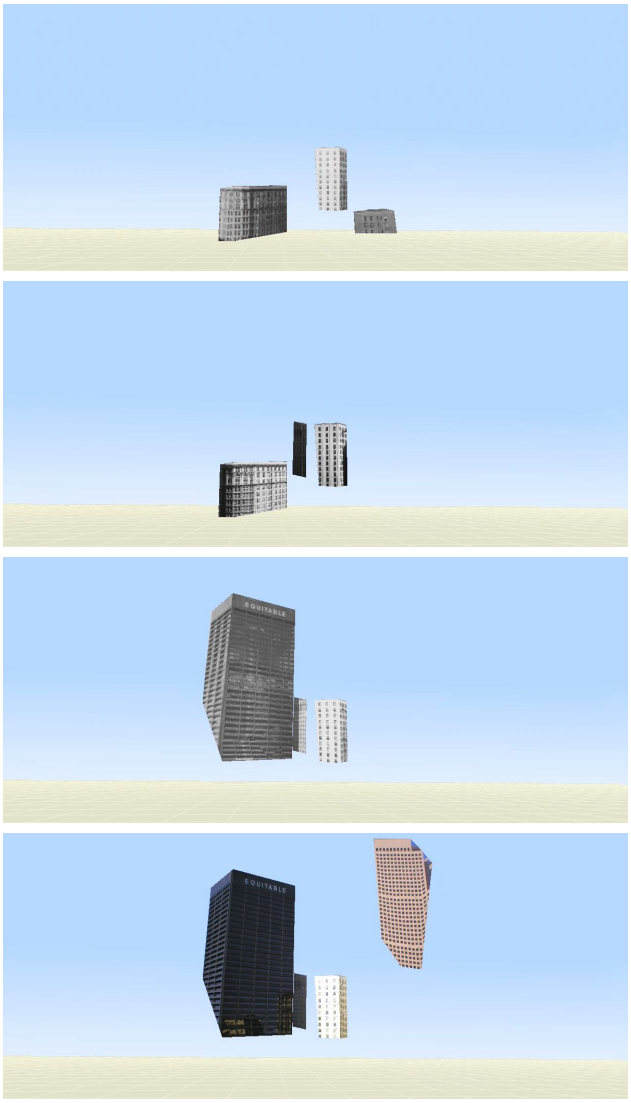


Figure 10. Time-varying 3D model. Here, we see the scene as it appeared at 4 different times from the same viewpoint. This result is generated automatically given 2D point correspondences across 6 unordered images as input. We perform SFM, determine occluding surfaces, compute the visibility matrix, solve the CSP using local search to infer temporal ordering, group points based on common dates of existence, compute 3D convex hulls, and texture triangles based on where they project into each image.

8. Conclusion

In this paper, we have shown that constraint satisfaction problems provide a powerful framework in which to solve temporal ordering problems in computer vision, and we have presented the first known method for solving this ordering problem. The largest obstacle to a fully automated system remains the variety of misclassifications enumerated above. In future work, we hope to extend the occlusion reasoning of our method to deal with occlusions by objects not explicitly modeled in the scene, such as trees, by considering image appearance around unmeasured projections of 3D points. In addition, with increasing numbers of misclassified points, structure segmentation decreases in quality, as fewer and fewer points have precisely the same dates of existence. We hope to find more robust methods of segmenting scene structure in order to automatically create high quality, time-varying 3D models from historical imagery.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0534330.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[2] S. Badaloni, M. Falda, and M. Giacomin. Integrating quantitative and qualitative fuzzy temporal constraints. *AI Commun.*, 17(4):187–200, 2004.

[3] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(3):61–95, May 1991.

[4] D. Dubois, H. Fargier, and P. Fortemps. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147:231–252, 2003.

[5] D. Dubois, H. Fargier, and H. Prade. Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. *Applied Intelligence*, 6:287–309, 1996.

[6] O. D. Faugeras, E. Le Bras-Mehlman, and J. D. Boissonnat. Representing stereo data with the Delaunay triangulation. *Artif. Intell.*, 44(1-2):41–87, 1990.

[7] M. Ge and M. D'Zmura. 4D structure from motion: a computational algorithm. In *Computational Imaging.*, pages 13–23, June 2003.

[8] D. Jelinek and C. J. Taylor. View synthesis with occlusion reasoning using quasi-sparse feature correspondences. In *Eur. Conf. on Computer Vision (ECCV)*, pages 463–478, 2002.

[9] D. D. Morris and T. Kanade. Image-consistent surface triangulation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 332–338, 2000.

[10] L. Sigal and M. J. Black. Measure locally, reason globally: Occlusion-sensitive articulated pose estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2041–2048, 2006.

[11] N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, pages 835–846, 2006.

[12] R. Sosic and J. Gu. 3,000,000 queens in less than one minute. *SIGART Bull.*, 2(2):22–24, 1991.

[13] C. J. Taylor. Surface reconstruction from feature based stereo. In *Intl. Conf. on Computer Vision (ICCV)*, page 184, 2003.