Social Snapshot: A System for Temporally Coupled Social Photography

Robert Patro, Cheuk Yiu Ip, Sujal Bista, and Amitabh Varshney ▪ *University of Maryland, College Park*

Social Snapshot actively acquires and reconstructs temporally dynamic data. The system enables spatiotemporal 3D photography using commodity devices, assisted by their auxiliary sensors and network functionality. It engages users, making them active rather than passive participants in data acquisition.

Since the invention of photography, taking pictures of people, places, and activities has become integral to our lives. In the past, only purposeful, precious moments were the primary subjects of photography. But technological advances have brought photography to our everyday lives in the form of compact cameras and even cell phone cameras.

The next phase in the photography revolution, 3D photography, can bring users together to socialize and collaboratively take pictures in an entirely new way. However, transforming a photographic scene from 2D to 3D requires introducing multiple images of the same underlying geometry from different viewpoints. The reconstruction of 3D geometry from multiple overlapping images is the classic structure-from-motion (SFM) problem in computer vision. Typically, the instruments used to acquire photographs are tediously calibrated to produce precise measurements.

To simplify 3D photography, our Social Snapshot system performs active acquisition and reconstruction of temporally dynamic data. Using multiple users' cell phone cameras and no preliminary calibration, it achieves approximate but visually convincing renderings of 3D scenes, even though the quality of such cameras is typically even lower than that of point-and-shoot devices.

Social Snapshot's Contributions

Social Snapshot's contributions fit naturally into two categories: technical and social.

The technical contributions are improved algorithms and techniques that enhance our system's novelty and scalability. For example, Social Snapshot produces a textured and colored-mesh reconstruction from a loosely ordered photo collection, rather than the sparse or dense point reconstructions produced by related approaches. In addition, it features locally optimized mesh generation and viewing. Finally, it provides camera network capabilities to support synchronized capture of temporally dynamic data.

The social contributions lead to a new way of thinking about the interplay between data acquisition and social interactions. They also let us define social photography as an active, rather than a passive, endeavor. For example, Social Snapshot encourages collaborative photography as a social endeavor, letting users capture dynamic action by synchronizing their photographs. It leverages social trends such as online media sharing and event organization to spur a novel data acquisition mode.

For a look at some of the previous research on which Social Snapshot is based, see the "Related Work in Scene Visualization and Computer Vision" sidebar on pages 78–79.

System Operations

Social Snapshot includes three main phases: acquisition, reconstruction, and display (see Figure 1).

 Published by the IEEE Computer Society 0272-1716/11/$26.00 © 2011 IEEE

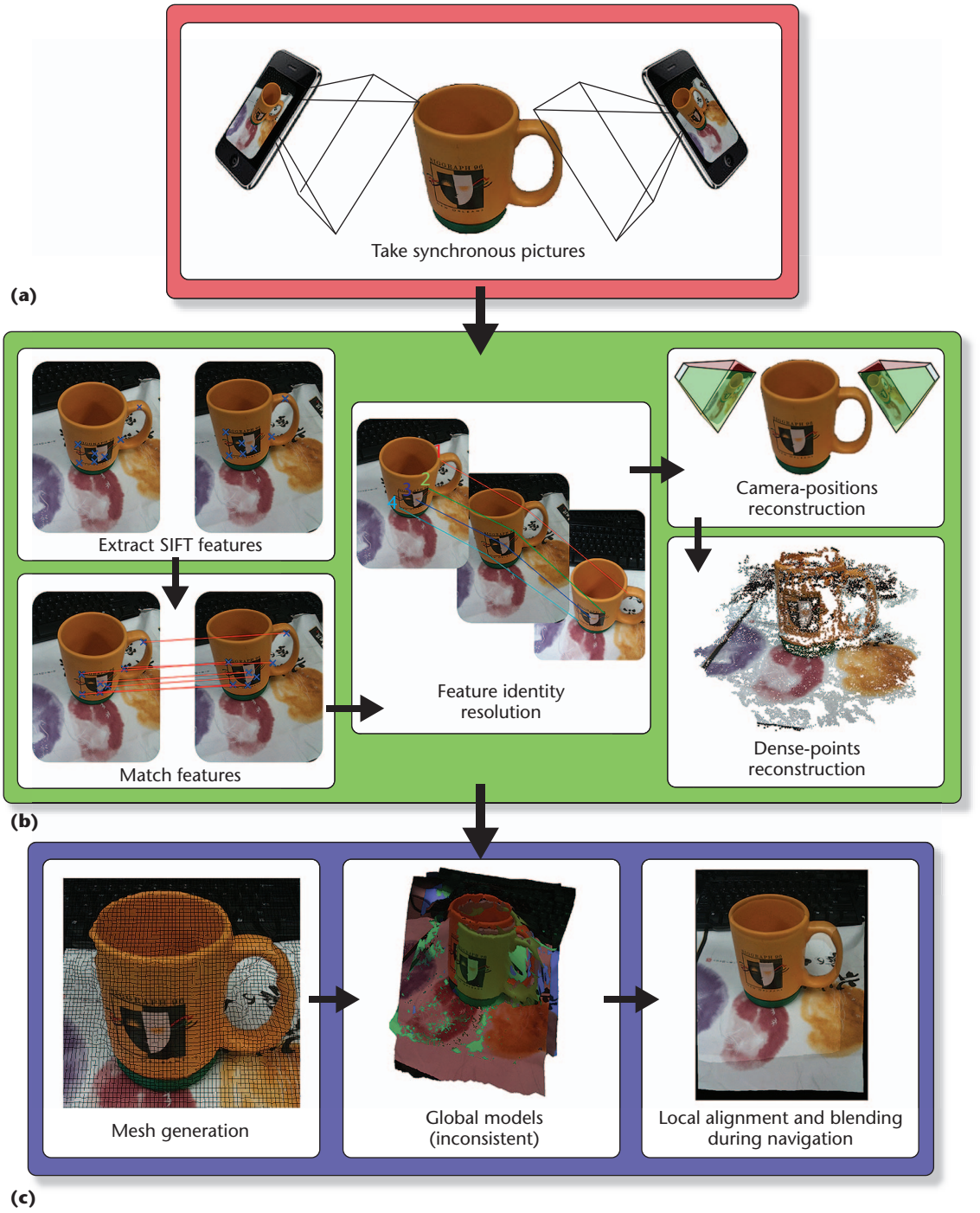


Figure 1. The Social Snapshot system comprises three phases: (a) During the acquisition phase, participants choose the scene they wish to capture and take synchronous photos of it. (b) During the reconstruction phase, the system uses the loosely ordered collection of photographs to construct a locally consistent 3D model of the scene at each moment in time. (SIFT stands for scale-invariant feature transform.) (c) During the display phase, users can freely navigate the spatiotemporal reconstruction.

Acquisition

Modern smart phones are sophisticated devices with many sensors and facilities. For example, this project's experimental platform, the iPhone 3GS, includes a 3-megapixel camera, accelerometer sensors, and Bluetooth connectivity. Phones from many other vendors (such as Symbian and Android phones) have similar configurations. The iPhone 3GS lens gives a field of view of approximately 60 degrees, which is considered wide in photography. This field of view facilitates capturing large parts of the scene, including overlaps with images captured by nearby cameras.

In this phase, participants use network-connected

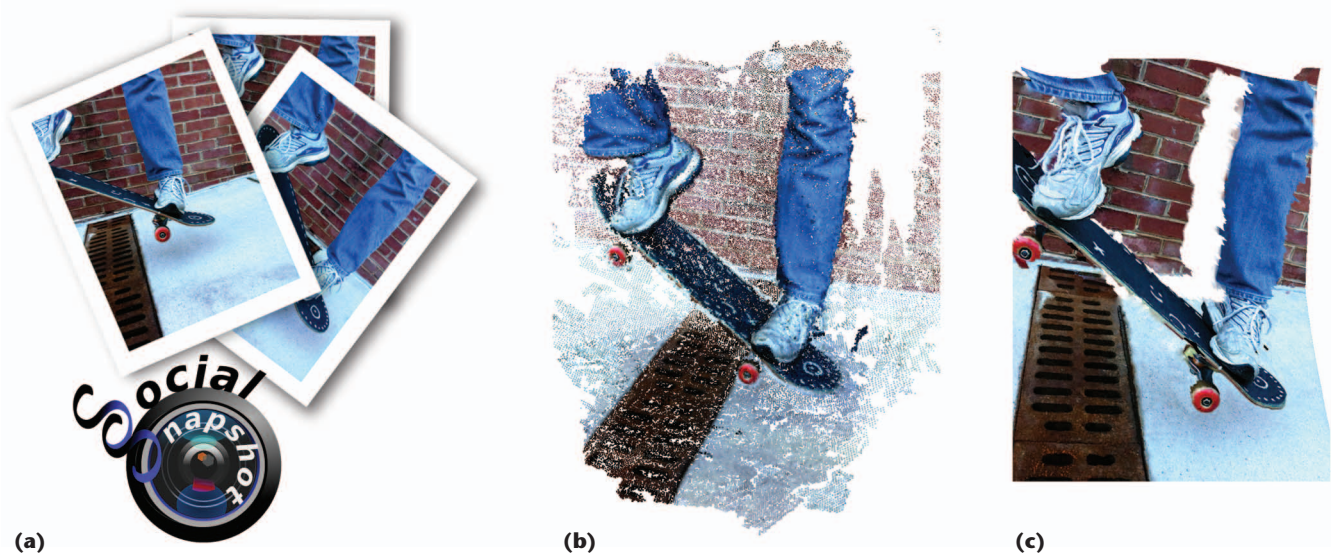


Figure 2. Social Snapshot enables the synchronized acquisition of highly dynamic data using commodity smart phones. (a) The system uses a series of synchronously collected photographs to determine the pose of the cameras employed for acquisition. (b) The system then uses this information to reconstruct a dense point cloud. Finally, the system generates several locally optimal meshed surfaces (one for each input camera). (c) This is one such surface. The viewpoint is altered significantly from that of the acquisition camera to illustrate the reconstructed surface and depth discontinuities. Capturing and reconstructing such dynamic data is only possible using a highly synchronized acquisition system.

Figure 3. During the acquisition phase, Social Snapshot connects the iPhones in a peer-to-peer network using Bluetooth. This network enables the synchronized acquisition of photographs among the participating devices. The system then uploads the resultant data to the reconstruction server using the broadband 3G network.

smart phones to acquire synchronized photographs of a scene (see Figure 2). Social Snapshot's social and collaborative aspects are driven by synchronization using these phones' network facilities. Because 3D reconstructions of time-varying scenes require multiple views, such synchronization is necessary. We achieve it by using the iPhones' Bluetooth connections to create a peer-to-peer network among the phones. One iPhone initiates connections to its nearby peers. The system aligns the phones' clocks using a Bluetooth message right before each acquisition session to optimize synchronization. After this alignment, one phone can trigger the other phones' cameras to take synchronized photographs. Each user employs his or her smart phone to take overlapping pictures from different viewpoints.

The system also records the phone's orientation during acquisition, inferring the phone's coordinate frame from the accelerometers and magnetometer. We could also use GPS to track location, but the current accuracy is far too coarse to be useful in the acquisition scenarios we consider in this project. Once the acquisition session is complete, the users can upload the photographs taken by the phones to the servers for reconstruction through the broadband 3G or Wi-Fi connections.

Figure 3 shows how Social Snapshot utilizes the iPhone's network capabilities. On average, the Bluetooth message's delay is only 30 milliseconds, with an 11-ms standard deviation. The system records this delay and offsets the clock alignment

appropriately to optimize the synchronization among the iPhones in the acquisition network.

Reconstruction

During reconstruction, the system transforms the loosely structured photographs captured during acquisition into a set of 3D models of the scene as it varies over time.

Let $\mathbf{P}_t$ denote an ordered set of photographs acquired by the system at time t, and let $\mathbf{P}_t^i$ be the ith photograph in this set. We use photograph $\mathbf{P}_t^i$ and image i interchangeably. If the acquisition subject is dynamic, we consider each set $\mathbf{S} = \mathbf{P}_t$ independently. Otherwise, we consider all sets $S = \{P_t\}_{t=0}^{T}$ of photographs taken during acquisition.

Feature extraction. We use scale-invariant feature transform (SIFT)[1] features because they're fairly robust and reliable. The feature extractor consumes an image i and produces a sparse set of features $\mathbf{f}_i = \{f_i^0, f_i^1, \ldots, f_i^n\}$. Each feature corresponds to a pixel location and provides a descriptor of the feature discovered at this location. The SIFT feature descriptor is a pair $(\Theta, \mathbf{v})$, where Θ is the feature's orientation, and $\mathbf{v} \in \mathbb{R}^{128}$ is a feature vector.

Putative feature matching. Once the system has extracted each image's features, we must find a set of correspondences between them to solve the SFM problem. Typically, the system extracts thousands of features for each image. So, when trying to find a putative set of correspondences between image features, we must use a method that lets us quickly determine whether a pair of features is a good candidate for further examination.

Let $\mathbf{f}_i$ be the set of features from image i. We consider each feature as a point in a high-dimensional space, and we build a query structure on the basis of a spatial, hierarchical decomposition of those points. Our system does this efficiently using ANN, a library for approximate-nearest-neighbor searching (www.cs.umd.edu/~mount/ANN). Given such a structure, we can consider each feature point not in $\mathbf{f}_i$ as a query point, and perform a proximity query on the structure we've built.

A pair of features $\left(f_i^x, f_j^y\right)$ constitutes a putative match if they reside close together in the high-dimensional feature space with respect to the other features. More specifically, we consider $\left(f_i^x, f_j^y\right)$ a putative match if $d(f_i^x, f_j^y) \leq \tau \times \min\left(d(f_i^x, f_j^z)\right)$, $z \neq x, y$, where τ is a threshold. We've observed that $\tau = 2/3$ produces experimentally good results, and this is what we've used in this article. Figure 4 illustrates this process. We denote the set of putative correspondences between two feature sets $\mathbf{f}_i$ and $\mathbf{f}_j$ as $\tilde{\mathbf{m}}_{ij}$. This process outputs a set of putative correspondences (some of which might be the null set) between each pair of images in the original dataset.

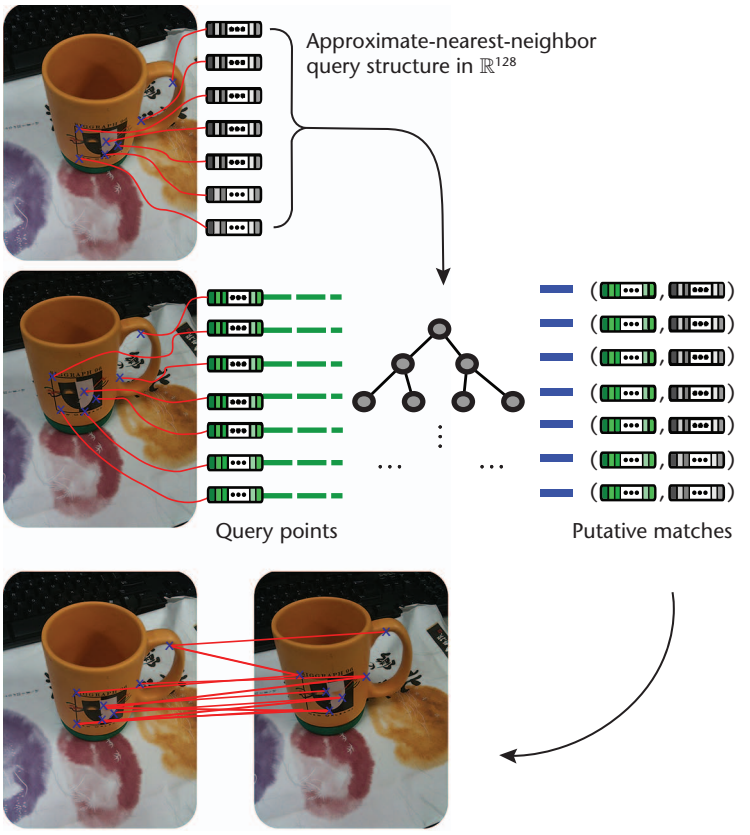


Figure 4. Treating scale-invariant feature transform (SIFT) features as vectors in a high-dimensional space lets us quickly determine putative matches by performing approximate distance queries. Such a filtering among the possible set of correspondences is essential to make the feature identity resolution problem tractable.

Inlier estimation. Although the putative-matching phase does a good job of filtering out highly improbable matches, each correspondence is considered in isolation and depends only on the orientation and feature vector corresponding to each feature. The true set of correspondences between two images will also obey the epipolar constraints. So, given $\tilde{\mathbf{m}}_{ij}$, we seek the fundamental matrix $\boldsymbol{M}_{ij}$ relating these two images, which explains the largest subset of $\tilde{\mathbf{m}}_{ij}$.

We compute $\boldsymbol{M}_{ij}$ using the eight-point algorithm.[2] To account for the presence of outliers, we perform this estimation in a Ransac (*ran*dom *sa*mple *c*onsensus) loop. For each pair of images i and j, $\hat{\mathbf{m}}_{ij}$ denotes the inlying correspondences. We define a correspondence as an inlier when it is explained by the transformation $\boldsymbol{M}_{ij}$. We also store the inlier ratio $\left(\#\hat{\mathbf{m}}_{ij} / \#\tilde{\mathbf{m}}_{ij}\right)$.

Related Work in Scene Visualization and Computer Vision

The Social Snapshot project combines collaborative photography and reconstruction and visualization of objects in 3D using the commodity cameras and sensors in mobile phones. The project draws heavily on previous research in scene visualization from computer graphics and structure from motion (SFM) in computer vision. Here, we briefly review some of this research.

Photo Tourism

The advent of digital cameras, coupled with the ability to upload photos and their corresponding metadata to the Internet, has integrated a convenient channel for taking digital pictures and a means of distributing them around the world. By collecting and combining photographs taken at the same place, such as popular tourist landmarks, we can view these locations from different perspectives and see how they look at different times of the day or even in different seasons. Photo tourism systems collect Internet tourism photos, find matching pictures, and organize them to give users a spatially driven, geometry-aware browsing experience.

For example, the University of Washington's community-photo-collections project and Microsoft's Photosynth reconstruct 3D feature points of architectural tourist landmarks for seamless browsing of multiple photographs.[1,2] James Hays and Alexei Efros filled in missing regions in photographs of a scene by using other photographs of the same place.[3] Xiaowei Li and his colleagues summarized scenes of landmarks with iconic scene graphs according to 2D appearance and 3D geometric constraints.[4] Paul Debevec and his colleagues modeled and rendered architecture photorealistically from a few photographs.[5]

The idea of organizing and structuring vast photograph collections using computer vision techniques seems intuitive, given the underlying data's highly spatial and visual nature. Meanwhile, the wealth and heterogeneity of available data have presented a constant source of new, interesting problems for this fledgling research field. However, despite the tremendous advances in recent photo tourism research, the underlying data—and hence the mechanism for acquisition and reconstruction—remains primarily static and passive. When the structure underlying the photo collection lacks temporal dynamism, this approach is usually sufficient. Other recent research explores the acquisition and reconstruction of time-varying scenes but relies on professional-grade equipment and a highly synchronized, calibrated camera network.[6]

Inertial Sensors

We enforce a gross alignment between inertial-sensor measurements and estimated camera pose to prevent the invalid pose estimation of cameras. Significant research has used inertial sensors to help solve SFM and related problems. The most recent relevant research in this area is that of Kihwan Kim and his colleagues, who used GPS readings to build density maps of urban areas, which they then combined to produce a coarse reconstruction of buildings.[7] Their article cites other relevant research.

View Interpolation

Noah Snavely and his colleagues' Photo Tourism system performs triangulated and planar morphs for view interpolation.[2] For triangulated morphs, it renders a mesh from different views and blends the rendered images on the basis of the camera position. For planar morphs, the system projects two views into a common plane; when the camera moves from one view to another, the system performs a cross-fade of the projected images. The Photo Tourism system also selects and warps images on the basis of the user's position; older images fade out while newer images are blended in.[8]

Instead of images from the different views, we blend local 3D models. We reconstruct dense point clouds of the scene and use an iterative constrained optimization routine to generate triangulated surfaces. Our system interpolates between local-mesh models on the basis of the viewer's direction and position.

Michael Goesele and his colleagues perform a global reconstruction, which is time-consuming but produces consistent results.[9] Conversely, Sameer Agarwal and his colleagues take a local approach, performing optimizations only for the images in a given window.[1] Social Snapshot op-

Feature identity resolution. Once we begin considering reconstructions from more than two images, we label each feature point with a unique identifier. If $\left(f_i^x, f_j^y\right) \in \hat{\mathbf{m}}_{ij}$ and $\left(f_j^y, f_k^z\right) \in \hat{\mathbf{m}}_{jk}$, then we label f_i^x, f_j^y, and f_k^z with one globally unique identifier. Each correspondence we discover simply suggests that two features from different images correspond to the same real-world point. We refer to the process of assigning each feature with a globally unique identifier as *feature identity resolution*.

We define $\boldsymbol{F}$ as the set of globally unique features. Formally, we wish to discover a map, ID: $\boldsymbol{F} \rightarrow \mathbb{Z}^*$, from the set of features to the set of nonnegative integers such that $ID\left(f_i^x\right) = ID\left(f_j^y\right) \Leftrightarrow \left(f_i^x, f_j^y\right) \in \hat{\mathbf{m}}_{ij}$. We also wish to impose the nondegeneracy constraint that no two features in an image should have the same identifier. We enforce this condition in the putative-matching phase by making the set of putative correspondences between a pair of images injective (that is, each feature in i can have only one putative correspondence in j, and vice versa).

timizes reconstructions locally, each with a group consisting of the primary camera and its nearest neighbors. This process is fast; the system handles the inconsistencies the process introduces by intelligently blending neighboring meshes.

References

1. S. Agarwal et al., "Building Rome in a Day," *Proc. IEEE 12th Int'l Conf. Computer Vision* (ICCV 09), IEEE CS Press, 2009, pp. 72–79.
2. N. Snavely, S.M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," *Proc. Int'l Conf. Computer Graphics and Interactive Techniques*, ACM Press, 2006, pp. 835–846.
3. J. Hays and A.A. Efros, "Scene Completion Using Millions of Photographs," *Comm. ACM*, vol. 51, no. 10, 2008, pp. 87–94.
4. X. Li et al., "Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs," *Proc. European Conf. Computer Vision: Part 1*, LNCS 5302, Springer, 2008, pp. 427–440.
5. P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. 23rd Ann. Conf. Computer Graphics and Interactive Techniques*, ACM Press, 1996, pp. 11–20.
6. A. Sankaranarayanan et al., "Modeling and Visualization of Human Activities for Multicamera Networks," *EURASIP J. Image and Video Processing*, vol. 2009, 2009, article 259860.
7. K. Kim et al., "Localization and 3D Reconstruction of Urban Scenes Using GPS," *Proc. 12th IEEE Int'l Symp. Wearable Computers* (ISWC 08), IEEE Press, 2008, pp. 11–14.
8. N. Snavely et al., "Finding Paths through the World's Photos," *ACM Trans. Graphics*, vol. 27, no. 3, 2008, article 15.
9. M. Goesele et al., "Multi-view Stereo for Community Photo Collections," *Proc. IEEE 11th Intl. Conf. Computer Vision* (ICCV 07), IEEE CS Press, 2007.

We solve the feature-identity-resolution problem by building a global feature graph $\mathbf{G_F}$ and assigning each feature the identifier corresponding to its connected component in the feature graph. To construct $\mathbf{G_F}$, we first assign a globally unique identifier $\tilde{id}\left(f_i^x\right)$ to each image feature. Next, we iterate over the set of all correspondences $\hat{M} = \cup_{i<j<C} \hat{\mathbf{m}}_{ij}$, where C is the number of cameras or the number of images taken at time step t. For each corresponding pair of features $(f_i^x, f_j^y) \in \hat{\mathbf{m}}_{ij}$, we add the edge $\left(\tilde{id}\left(f_i^x\right), \tilde{id}\left(f_j^y\right)\right)$ to $\mathbf{G_F}$.

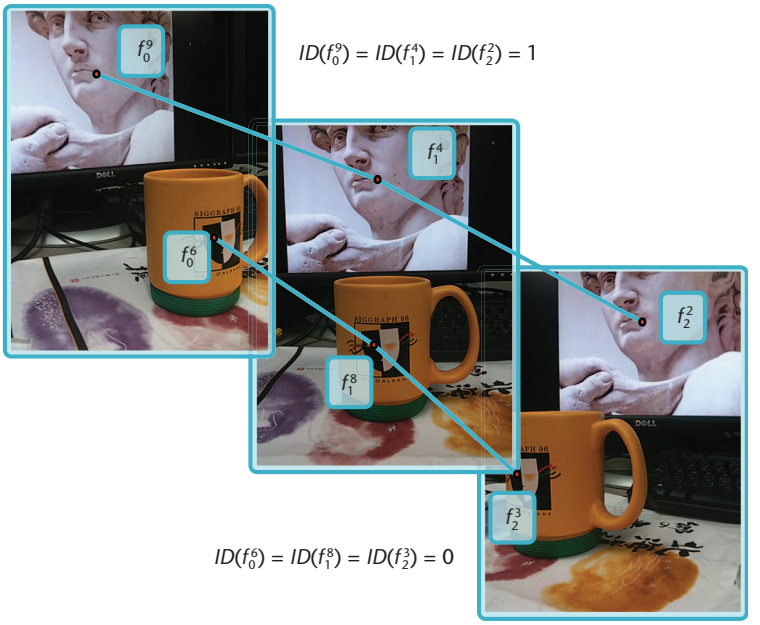


Figure 5. Feature identity resolution de-aliases the feature points f_i^x across multiple images and assigns each feature a unique identifier. Having unique identifiers for referring to feature points greatly simplifies the reconstruction algorithm.

Finally, we perform a connected-component analysis on $\mathbf{G_F}$. Each connected component consists of edges representing a sequence of correspondences between image features. The correspondence relation's transitivity ensures that every vertex in a given connected component is posited to refer to the same real-world feature. To resolve the global feature identifiers, we assign a unique identifier to each connected component of $\mathbf{G_F}$, and we let each feature's identifier equal the identifier of the connected component containing it. Figure 5 illustrates this concept.

Iterative SFM. We've built a custom iterative SFM engine atop the Basic Image Algorithms Library (BIAS; www.mip.informatik.uni-kiel.de/~wwwadmin/Software/Doc/BIAS/html/main.html). We begin the SFM computation by determining the essential matrix between an initial camera pair because we have a good estimate of the intrinsic camera parameters.

The first camera of the pair is at the origin, with its local frame aligned to the canonical axes of $\mathbb{R}^3$. The essential matrix gives the second camera's relative position. Once the system has reconstructed the initial camera pair's pose, we estimate the world-space positions of their matching feature points. The system adds those points with a low reprojection error to the global set of reconstructed features $\mathbf{R}$. $\mathbf{R}_c$ denotes the set of cameras that have contributed to the reconstruction of features in $\mathbf{R}$.

The pipeline's iterative phase, which poses the remaining cameras, runs in rounds. We denote the current round as r. The system ranks all cameras according to their features' overlap with $\mathbf{R}$. Let $\mathbf{f}_i$ denote the set of features in camera i, and let $\mathbf{ID}_i = \left\{ID(f_i^x)\right\}_{f_i^x \in \mathbf{f}_i}$ denote the set containing those features' global identifiers. To rank the cameras, we consider the cardinality of $\mathbf{o}_i = \mathbf{ID}_i \cap \mathbf{R}$. The system considers cameras with high overlaps for pose estimation in round r. For each considered camera, i, the system gives a robust linear estimation of its pose by the direct linear transform in a Ransac loop,[2] considering the 2D-to-3D correspondences encoded in $\mathbf{o}_i$.

We've developed a novel meshing procedure that uses an iterative constrained-optimization routine inspired by a mass-spring system.

When acquiring a dataset, the system is typically provided with enough photographs so that, in terms of reconstruction quality, leaving a camera unposed is less costly than incorrectly posing it. To help avoid grossly incorrect camera poses, we exploit the inertial-sensor data collected in concert with each photograph. This data consists of the accelerometer, magnetometer, and GPS readings, which provide a reasonable orientation and highly approximate position for the device. If the computed camera pose significantly disagrees with the inertial-sensor measurements, the system considers the reconstructed pose invalid. In this case, we postpone the determination of this camera's pose until a later round, when hopefully more overlapping feature points will have been reconstructed. However, if we can't determine a valid pose that at least grossly agrees with the inertial-sensor measurements, we leave this camera out of the reconstruction.

Next, the system considers any corresponding features between i and all the cameras in $\mathbf{R}_c$, and triangulates the world-space position of the valid features that aren't already in $\mathbf{R}$. The system then adds these newly reconstructed points to $\mathbf{R}$, and we perform the same procedure for the rest of the cameras selected for pose estimation in round r. Once the system has posed all the selected cameras, it advances to round $r + 1$.

The iterative estimation terminates if there are no more cameras left to pose or if the remaining cameras have too few overlaps with $\mathbf{R}$. The system optimizes the estimated camera poses one final time, using sparse bundle adjustment.[3]

Local scene reconstruction. We construct locally optimized texture-mapped 2.5D dense point clouds and meshes for each image. Given the limited number of synchronously acquired images, we can't construct a globally consistent model. We use a few (two or three) of the neighboring images to reconstruct the model for each camera. Because the images cover only a small area, the reconstructed camera models are locally consistent. Our goal is to enable seamless navigation without a globally consistent model.

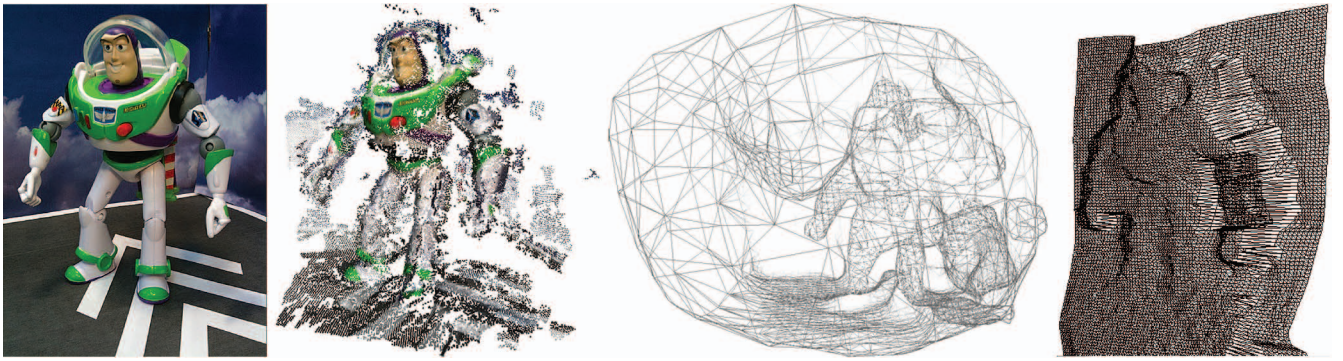
Next, we find a posed camera's closest neighbors for reconstruction. Let $\mathbf{C}$ denote the set of posed cameras. The closest neighbor for a given camera i is $i_0 = \arg\max_{j \in \mathbf{C}} \#\left(\mathbf{ID}_i \cap \mathbf{ID}_j \cap \mathbf{R}\right)$, the second closest neighbor is $i_1 = \arg\max_{j \in (\mathbf{C} \setminus \{i_0\})} \#\left(\mathbf{ID}_i \cap \mathbf{ID}_j \cap \mathbf{R}\right)$, and the higher-order neighbors follow similarly.

We reconstruct dense point clouds using PMVS (Patch-Based Multi-view Stereo Software).[4] The 3D point clouds are practically 2.5D because the neighboring cameras have similar viewing directions and take pictures of only one side. Each point in this dense reconstruction maintains an average color, a corresponding visibility map, and a corresponding pixel location in each photograph that contributed to the dense point model. This reconstruction's locality ensures that the images map consistently to the points.

Because the captured photographs represent the true images from each reconstructed camera position, the system creates simple 2.5D depth meshes to assist in rendering and local 3D navigation. The construction of a globally consistent mesh is prone to accumulate tiny errors, contributed from each reconstructed camera. So, we instead exploit the images and locally optimized camera positions to generate optimal meshes for each camera position.

Many point cloud meshing techniques, such as Poisson surface reconstruction, generate only watertight surfaces. But these aren't suitable for generating 2.5D view-dependent meshes from our point clouds, which contain numerous discontinuities. Figure 6 shows how Poisson surface reconstruction adds artificial points to close up the surface when meshing the point cloud.

We've developed a novel meshing procedure that uses an iterative constrained-optimization routine inspired by a mass-spring system (see Figure 7). The meshes are 2.5D triangulations of dense points with respect to each camera. For an input image with dimensions $W \times H$, we create a spring



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

Figure 6. View-dependent mesh construction: (a) the captured image; (b) the constructed dense point cloud; (c) the Poisson surface reconstruction, which adds artificial vertices to generate a closed surface when the dense point cloud contains discontinuities; and (d) the mass-spring-inspired mesh surface reconstruction, which connects the discontinuities with respect to the image grid while maintaining the original vertices.

(a) (b) (c)

Figure 7. We use the mass-spring system to compute a locally optimized mesh for each camera: (a) the initialized mass-spring system, (b) the result of the mass-spring mesh optimization, and (c) the mass-spring mesh after final realignment. The resulting mesh closely matches the original image when viewed from the camera position.

system with $w \times h$ particles, where $w << W$ and $h << H$. The system aligns the particles at the spring system's corners with the input image's world coordinate position computed by the SFM algorithm. The spring system's remaining particles are uniformly spaced between the corner particles.

Next, we use the recovered camera parameters and construct a virtual camera. We then project the visible 3D points from the dense point cloud and the spring particles onto the virtual camera's view plane. For each spring particle, we find the closest projection of the visible 3D points from the dense point cloud that's within a radius k. If the system discovers such a point, the spring particle is associated with it. The system divides spring particles into three sets. Set A contains all the spring particles that have associated 3D points from the dense point cloud. Set B contains all the spring particles at the corners. Set C contains the rest. The system replaces the position of each spring particle in A with its associated 3D point from the dense-point-cloud data, which contains structural information on the photographed objects in world coordinates.

We then iteratively run the mass-spring optimization, keeping spring particles in A and B static while moving the particles in C. We stop the optimization once the mass-spring system stabilizes. Finally, we realign the generated mesh by considering the optimization constraints' camera parameters and location. The realignment doesn't change the distance between the spring particles and the virtual camera. The system locally optimizes each resulting mesh for its corresponding camera. During navigation, the system selects the displayed mesh according to its acquisition and viewing parameters.

Display

During this phase, users can navigate in space and time to explore the reconstructed scene in 3D,

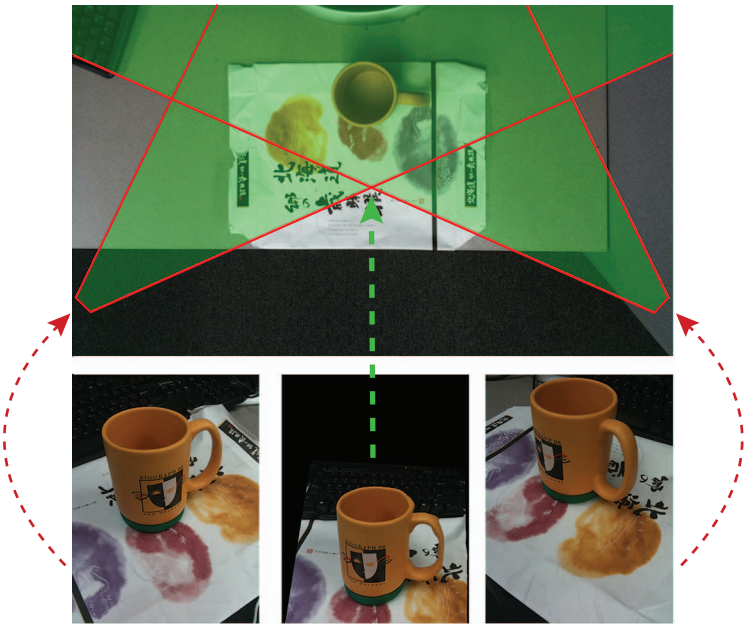


Figure 8. Users can move freely in the region marked in green. The system calculates the marked region using the acquisition cameras' view frustums. Using images from the initial photographs, the system generates and textures a locally optimal mesh. The bottom left and the right images show the actual photographs; the bottom center image is the rendered mesh for navigation.

constrained only by the available data. Here, we describe how our system handles navigation and the efficient display of our locally optimized meshes.

Model stabilization (image and object space). During rendering, the system selects and displays the mesh that's optimal with respect to the current camera. From the reconstruction stage, we have one 3D mesh for each acquisition camera. Some inconsistencies will exist between these local meshes. To overcome these inconsistencies and get the best possible view for each frame, the system prioritizes these meshes on the basis of the view and acquisition cameras' parameters. This prioritization favors the acquisition cameras with viewing directions and positions similar to those of the view camera. The system uses the mesh corresponding to the best camera from this ranking to show the final model. When a movement occurs, the system blends the previous and current best meshes over several frames to provide a smooth transition.

Navigation. We allow two forms of navigation: automated flythrough and constrained navigation. Automated flythrough creates a camera path that passes through all the initial cameras used to create the view-dependent mesh. The system conducts a smooth interpolation from one camera to the next. This navigation mode lets users see the final generated model from each initial camera's perspective. The system can further optimize this navigation mode by accepting user-generated paths.

Constrained navigation lets users move around the model with some limitations (see Figure 8). When several local meshes are blended together, gaps will be present because we're compositing local meshes that aren't fully consistent. Costly global mesh generation is necessary to make these meshes consistent. To minimize the gaps seen by the user, we implement a constrained 3D navigation model. The camera may move only inside the acquisition cameras' view frustums. Inside each frustum, the camera can rotate as long as it can see the model and as long as the dot product of the navigating camera's and original camera's view directions is greater than zero. These restrictions ensure that the scene is mostly visible in every frame, and they minimize visibility disocclusions.

Results

Here, various reconstructed scenes highlight the system's capability to synchronously capture photos and illustrate the quality of the locally reconstructed textured meshes.

We built Social Snapshot with performance in mind. At the same time, we're dealing with datasets that differ considerably from the typical datasets in recent large-scale SFM research. In particular, our underlying acquisition mechanism is active and time dependent, so we can use both spatial and temporal exclusivity to segment our data. Thus, the system can process each time step from an acquisition session almost completely in parallel, and this processing's results need not be reunited until the viewing phase. This characteristic makes our system highly scalable.

Nevertheless, for each dataset we present, we executed all the pipeline phases through the final camera pose estimation on a single desktop with 4 Gbytes of RAM and a 3-GHz Intel Core 2 Duo, in under 30 minutes. We performed the dense point reconstructions in parallel on a cluster of 25 machines, each with two quad-core 2.5-GHz Intel Xeon CPUs and 16 Gbytes of RAM. For each dataset, the dense point reconstruction of all of its constituent time steps required less than 5 minutes. Our viewer runs interactively (at approximately 100 Hz).

The Jumping Skater Dataset

This dataset demonstrates a skateboarder performing a trick called an Ollie. We captured the images leading up to and during the jump. To allow for reconstruction, it's important to have numerous cameras capture the subject at almost exactly the

same time. This reconstruction illustrates that our acquisition system, even though based only on commodity smart phones, is highly synchronized.

Figure 9 shows the reconstructed point clouds and meshes. Each segment of this beautifully executed jump was captured using 24 cameras; the dense point reconstruction for each time step contained an average of 77,896 points.

Star Trek *"Amok Time" Scene*

We recreated the fight scene from the "Amok Time" episode of *Star Trek* using action figures and captured the scene as a stop-motion animation sequence. Stop-motion animation provides a fun way to synthesize creative narratives. Social Snapshot uses numerous cameras to augment the animation with a new dimension.

Figure 10 shows the reconstruction of two sample time steps. We reconstructed each time step using 13 cameras; each time step's dense point set contained an average of 66,120 points.

Social Snapshot presents several interesting venues for future research. In particular, high-quality inertial sensors on new devices would let us rely less on costly global optimization procedures and more on direct sensor readings. In particular, the newest generation of smart phones, such as the iPhone 4, include both an accelerometer and a gyroscope. This makes it possible to delineate between rotational and linear acceleration, and therefore to track the motion of the phone. The ability to perform such tracking hints at the possibility of more efficient reconstruction algorithms. For example, it might be possible to use an iterative SFM procedure to pose the cameras only in the first acquisition time step, with the camera poses in all remaining time steps tracked and updated through the inertial-sensor readings. It's also likely that the ever-growing array of sensors will enjoy a synergistic relationship with the reconstruction algorithms to improve not only the reconstructions' efficiency but also their accuracy.

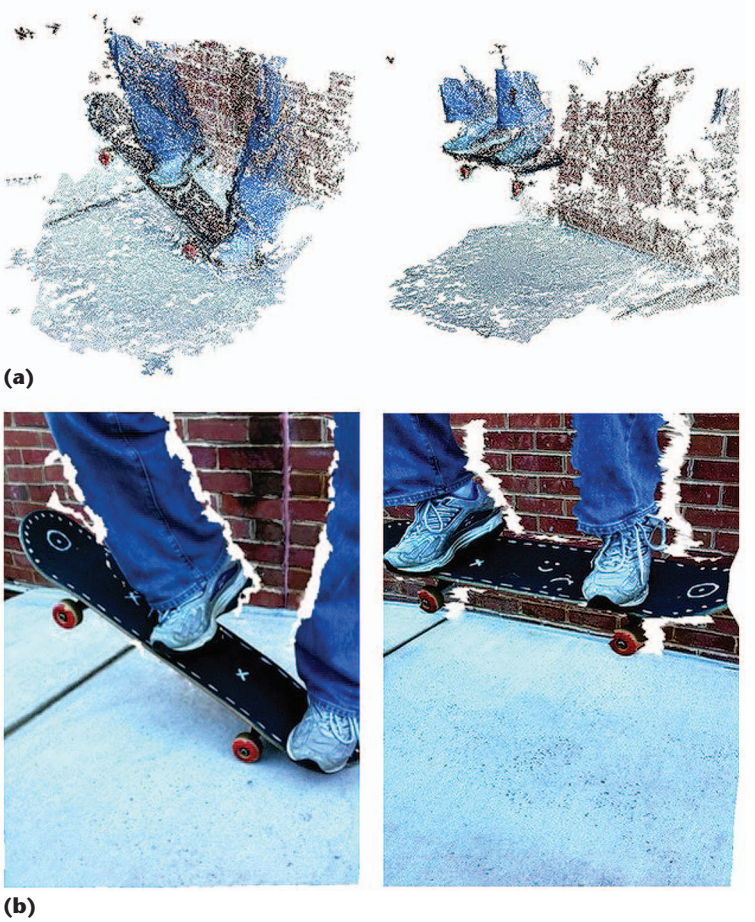


Figure 9. Reconstructing (a) point clouds and (b) meshes from the jumping skater dataset, for the skateboard taking off and the jump's apex. (Figure 2 shows an intermediate frame for this same dataset.) This figure illustrates the successful reconstruction of a highly dynamic underlying dataset. Such a reconstruction wouldn't be possible without the active synchronization mechanism, which is an integral part of the Social Snapshot system.

Acknowledgments

This research has been supported partly by US National Science Foundation grants CCF 04-29753, CNS 04-03313, CCF 05-41120, and CMMI 08-35572. We also gratefully acknowledge the support provided by the Nvidia CUDA (Compute Unified Device Architecture) Center of Excellence Award to the University of Maryland. Any opinions, findings, conclusions, or recommendations expressed in this article are the authors' and don't necessarily reflect the research sponsors' views.

References

1. D.G. Lowe, "Object Recognition from Local Scale-Invariant Features," *Proc. 7th IEEE Int'l Conf. Computer Vision* (ICCV 99), vol. 2, IEEE CS Press, 1999, pp. 1150–1157.
2. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge Univ. Press, 2004.
3. M.A. Lourakis and A.A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, 2009, article 2.
4. Y. Furukawa and J. Ponce, "Accurate, Dense, and Robust Multi-view Stereopsis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, 2010, pp. 1362–1376.

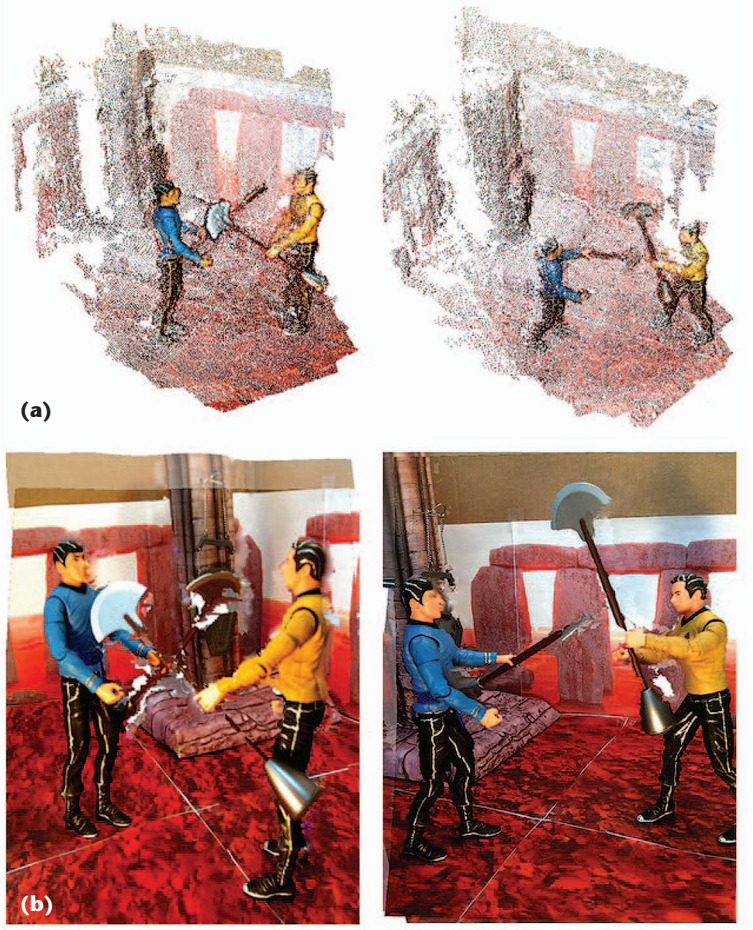


Figure 10. Reconstructed (a) point clouds and (b) meshes for two time steps of a stop-motion-animation sequence based on the "Amok Time" episode of *Star Trek*. Despite the fact that each reconstructed model is created using only a small subset of the cameras, the resulting renderings are visually convincing.

Robert Patro *is pursuing a PhD in computer science at the University of Maryland, College Park. His research interests include computational biology, scientific visualization, and computer graphics. Patro has a BS in computer science from the University of Maryland, College Park. Contact him at rob@cs.umd.edu.*

Cheuk Yiu Ip *is a pursuing a PhD in computer science at the University of Maryland, College Park. His research interests include computer graphics and 3D shape acquisition and comparison. Ip has an MS in computer science from Drexel University. Contact him at ipcy@cs.umd.edu.*

Sujal Bista *is pursuing a PhD in computer science at the University of Maryland, College Park. His research interests include computer graphics and human-motion understanding. Bista has an MS in computer science from the University of Maryland, College Park. Contact him at sujal@cs.umd.edu.*

Amitabh Varshney *is a professor of computer science and the director of the Institute for Advanced Computer Studies at the University of Maryland, College Park. His research focuses on exploring the applications of interactive and high-performance graphics and visualization in engineering, science, and medicine. Varshney has a PhD in computer science from the University of North Carolina at Chapel Hill. He's an IEEE Fellow. Contact him at varshney@cs.umd.edu.*

cn Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.