Swipe Mosaics from Video

Malcolm Reynolds, Tom S. F. Haines and Gabriel J. Brostow

Abstract—While a panoramic image mosaic is an attractive visualization for viewing many overlapping photos, its images must be both captured and processed correctly to produce an acceptable composite. We propose Swipe Mosaics as an interactive visualization that places the individual video frames on a 2D planar map that represents the layout of the physical scene. Compared to traditional panoramic mosaics, our *capture* is easier because the user can both translate the camera center and film moving subjects. Our *processing* and display also degrade gracefully if the footage lacks distinct, overlapping, non-repeating texture. Our proposed visual odometry algorithm takes an image pair and produces a distribution over (x, y) translations. Inferring a *distribution* of possible camera motions allows us to better cope with parallax, lack of texture, dynamic scenes, and other phenomena that hurt panoramic image mosaics or deterministic reconstruction techniques. The visual odometry regressor gains this robustness by training on renderings of synthetic scenes with known camera motion. We show that Swipe Mosaics are easy to generate, support a wide range of difficult scenes, and are useful for documenting a scene for closer inspection.

1 Introduction

The appeal of Microsoft's Photosynth [1] is just one indication of how people wish to capture environments for later navigation. Intuitive interactive navigation of video frames can mean more than just playing them back in temporal order. Users often wish to navigate the captured footage spatially. For certain dynamic scenes, the works of [2], [3], [4], and [5] explored *direct manipulation* of video. With interesting variations, those algorithms mapped a user's click-and-drag stroke to a sequence of frames elsewhere in the timeline. The location of the click and the direction of the mouse indicated to the system which pixels and what point or optical flow trajectory to query for in the whole sequence. We seek a similar direct user interaction for spatial navigation of scenes, which preserves the film's points of view and the veracity of the images. For example, imagine needing to inspect the gold handbag in Fig. 1 to place a bid in an online auction, or record scratches after a car accident. Our system allows casually captured video footage of the subject to be automatically converted, under some simple assumptions, into a navigatable "Swipe Mosaic".

When footage contains texture allowing for accurate estimates of optical flow, or appropriate interest points that allow for 2D or 3D pose estimation, then Image Based Rendering (IBR) techniques can be used to composite static or dynamic mosaics as set out by [6], or with the vast majority of crowd-sourced images, browsing the images in 3D as in [7] or [8]. Many everyday scenes lack texture, or otherwise break the assumptions made by current IBR and direct video manipulation systems. Our proposed IBR approach deals with failing pose estimation more gracefully than other methods in difficult scenes, yet the rendering quality and user interaction do not suffer adversely. Towards the objective of intuitively navigating video frames on ubiquitous devices, we present the following contributions:

- A regressor model trained using a simple graphics engine, that learns the relationship between image pairs and their 2D Euclidean transform parameters.
- A layout method that uses the probabilistic pairwise predictions from the regressor to produce a 2D location for each image, and then tries to detect and re-optimize loop-closures.
- A Swipe Mosaic interface, shown in Fig. 1, to display the video frames, allowing the user to perform content centric navigation and inspection by "swiping" scene elements. The interface can run as either a native application or on a web browser, allowing usage on smartphones.

In contrast to regular panoramic image mosaicing approaches, our system can analyze and visualize handheld camera footage with parallax, blur, textureless and specular areas, and moving subjects, with the visualization quality degrading gracefully in the case of especially difficult scenes.

2 Related Work

Since the genesis of Image Based Rendering for synthetic data [9], steady progress has been made toward beautiful and useful renderings from footage of the real world. Footage usually comes from multiple viewpoints, so progress is inherently dependent on having accurate estimates of relative camera poses. Here we summarize the most relevant interactive IBR approaches, starting with techniques for estimating the needed camera parameters.

Camera Poses: A comprehensive summary of methods for converting video frames into planar and cylindrical mosaics is presented in [10], while [11] cover spherical mosaics. They explain how stitching an image mosaic is easiest when all the images can be related to each other by homographies. This relation can exist when the camera is translated parallel to a planar scene, or when

• *The authors are with University College London.*

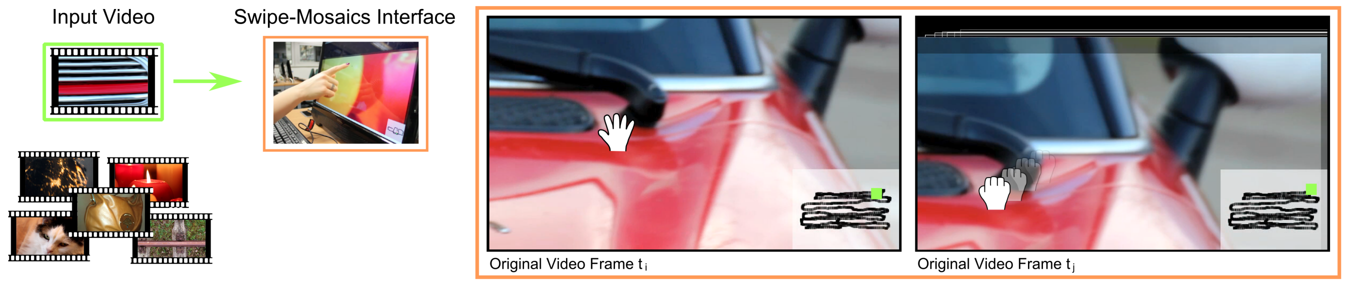


Fig. 1: Left: input videos containing "difficult" phenomena are used as inputs to our system. Right: the Swipe Mosaic interface allowing navigation over an image sequence. We propose Swipe Mosaics as an algorithm and associated interface which composites video frames into a content-centric navigable visualization. Some videos can already be browsed *spatially* by using existing mosaicing or IBR methods, our system broadens the range of usable videos because it is trained to tolerate scene motion, parallax, repeated structure, and lack of texture.

undergoing pure rotation. Szeliski also motivates and demonstrates robust ways of registering images to each other without matching detected interest points, such as through coarse-to-fine matching and phase correlation. Such registration benefits from either manual or interest-point based initialization, and assumes that the scene is textured. Textured scenes ensure convergence when minimizing the residual difference in the intensities of overlapping pixels. Texture can also help when mosaicing an image sequence, because optical flow is strongly correlated with visual odometry [12]. [13] show that estimating camera motion and warping to enforce a consistent parallel optical flow direction allows one to combine columns of pixels onto a 2D manifold, and not necessarily onto a planar, cylindrical, or spherical mosaic. Optical flow estimates are most accurate when the scene is textured, and [14] have a helpful system to compute the uncertainty of the estimated (u, v) flow components. We too benefit from texture in the scene, but are less reliant on it.

Initializing camera poses can be difficult in practice, even in textured scenes. Hardware attached to the camera can help [15], as demonstrated by [16] who fused visual cues with gyroscope data and [17] who used an inertial sensor to mitigate blur. [18] actively controlled the camera pose using a motorized telescope mount to stitch mosaics of thousands of photos. There are numerous other hybrid systems which fuse other data with images, but even [18], [19], and the Photosynth App [1] rely on interest point matching to register their images. The SIFT detection and features of [20] remain the standard by which interest point detection and matching is measured [21]. Finding enough matching interest points in an image collection means that photos can be registered to each other, adjusted for exposure, and blended into a large mosaic [22]. At least four points must be matched to compute the projective transform between two images, but in practice 10's and 100's of points are used with RANSAC [23] to robustly calculate an answer. The same approach and inflated number of distinct interest points is normal for estimating the translation and rotation of the 2D Euclidean transform, even though two corresponding points is enough, and solutions with corresponding lines and curves also exist [24]. The key issues are that large areas of real images have light or sparse texture, and that seemingly corresponding points may not represent the same 3D point in the world because of scene motion, motion blur, reflection, or repeated structures [25].

When building mosaics or other IBR and multi-view scene models, camera pose estimation is overwhelmingly seen as a self-contained problem. Even [26], whose system was designed to cope with moderately-sized moving objects and rotation-only cameras, performs global optimization by treating all the estimated pairwise camera-transforms as equally good. In contrast, our regressor (§3.1) reports high uncertainty for less textured or more dynamic scenes, and the subsequent layout computation (§3.2) incorporates this uncertainty. Swipe Mosaic visualizations can better cope with difficult (though typical) footage because we work with distributions rather than committing too early to interest-point matches or specific Euclidean transform parameters.

Probabilistic distributions on locations have been applied before, such as to help a "teleporting" robot with a range sensor localize itself in a known floorplan [27]. Probabilistic models are increasingly employed in Structure from Motion (SfM) too [28]. SfM classically requires running RANSAC over more suggested interest-point matches than the Euclidean transform (five are needed at minimum). SfM then estimates 3D camera poses and 3D scene point locations, and finally optimizes these estimates globally using repeated steps of Bundle Adjustment (BA) [24]. The stages of SfM are normally deterministic and notoriously computationally expensive, but we are particularly inspired by the recent work of [29] who use a less costly optimization to compute an initialization for a single iteration of BA. They convert the deterministic pairwise estimates to probabilistic constraints on a graphical model, which they solve with Loopy Belief Propagation [30]. The probabilistic approach gives a principled method of incorporating other information, such as geotags. Instead of replacing the final half of the BA pipeline with a probabilistic system, we propose to model pose probabilistically from the beginning.

Rendering & Interaction: Much like the direct manipulation works mentioned already and our own interface, Dynamic Mosaics [31] prominently display for interaction a current frame from the input footage. Their rendering method occupies a middle ground between ours and that of classic image mosaics, in that they dynamically stitch onto that frame *some* spatially neighboring frames, choosing neighbors which share a large number of inlier correspondences. This obviously limits the variety of scenes which can be displayed, so they have an alternate mode based on the similarity transform, which requires somewhat fewer correspondences. We require no explicit correspondence points.

Interest-point based registration with subsequent Bundle Adjustment has allowed numerous interesting IBR prototypes to emerge. Panoramic Video Textures (PVT) [32] register and play video clips inside an otherwise static cylindrical panorama. A competing PVT system [33] allows parts of the XYT-volume to be played back in different order, *e.g*. making explosions look like implosions. Also reliant on interest point matches but with an alternative optimization to BA, [34] are able to stabilize shaky videos to follow different target trajectories.

The Lumigraph [35] and Light Field Rendering [36] cleverly allow the user to recombine the rays captured by an array of cameras. Interfaces allow users to navigate the plenoptic function spatially, and to simulate new focal lengths. [37] showed a hardware based system for capturing a reduced-size 3D plenoptic function. The recent system of [38] massively simplifies the process of capturing light fields by giving fast feedback about what parts of the static scene have been adequately filmed. They employ the PTAM [39] real-time SfM system which registers their cameras if enough interest points are available, and the camera does the characteristic “SLAM wiggle” [40].

[41] discuss the differences between strip panorama systems, and propose a multiviewpoint panorama which stitches together large regions of photos that were shot with a hand-held camera. The strength of their interface is that users can override the stitching to (de)emphasize perspective effects in different parts of the scene. Their system relies on the Bundler SfM system [7] for camera registration. The Street Slide system of [42] shows another interface to multiviewpoint panoramas, which was part of our motivation for a 2D interface. The Photo Tourism work of [7] and [43] was instrumental both for releasing Bundler and the insight that sufficiently large photo collections could be browsed in 3D. When images show the same objects or objects in-the-round, the viewer’s transitions are rendered smoothly, and [8] offer especially smoothed transition effects for images that are very far apart in 3D. These systems prefer to cull low-texture and low-quality images, and endeavor to eliminate moving objects from their collections. In contrast, our users are filming video of something specific for interaction in a 2D swipe interface, need *that* sequence to work, and may not have the benefit of static scenes and distinct interest points.

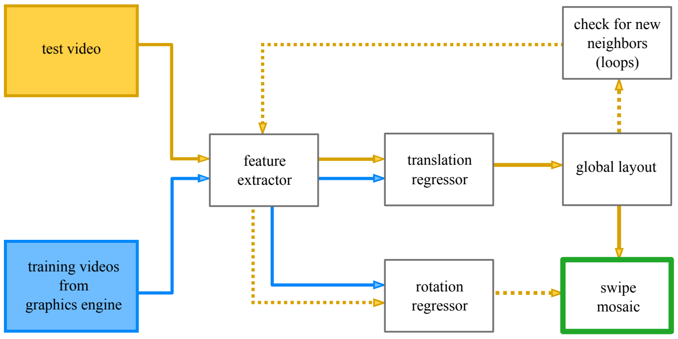


Fig. 2: System diagram illustrating how a video is analyzed to generate a Swipe Mosaic. Blue lines indicate the offline training process. Dotted yellow lines indicate post processing steps, which take place after an initial layout is found.

3 Swipe Mosaic Construction

Our system takes as an input a video sequence or temporally ordered set of images $\{\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_N\}$ and presents them in a new type of interactive mosaic. Valid inputs to our system include scenes which could be used to create a panoramic mosaic, but also include scenes containing significant parallax and dynamic objects, so the Swipe Mosaic avoids trying to stitch all the inputs together seamlessly. As an overview of our approach, we first select pairs of images and make predictions of the *relative* camera motion for each pair, before combining those predictions using a global least squares optimization. The predictions form a distribution over possible camera motions. The layout algorithm locates the images on a 2D manifold so they can be visualized using our Swipe Mosaic interface. Finally, several postprocessing steps may be performed to further improve the viewing experience. The overall pipeline of our visual odometry regressor and layout system is shown in Fig. 2.

Pair selection generates a set $\mathcal{P} = \{(j_1, k_1), \ldots\}$, following which camera motion will be estimated between image pairs $\{(\boldsymbol{I}_{j_1}, \boldsymbol{I}_{k_1}), (\boldsymbol{I}_{j_2}, \boldsymbol{I}_{k_2}), \ldots\}$. A number of strategies can be employed to select pair indices - some selection is necessary as comparing $O(n^2)$ pairs is computationally infeasible for large sequences. It is possible to anticipate loops in the ordered set by finding image pairs which are temporally distant but show the same location. Possible techniques for modeling such similarity include SIFT matching [20], GIST scene descriptors [44], simple L2 intensity distance, or geodesic distance models such as Isomap [45]. We evaluated these methods but achieved superior results by initially picking only close temporal neighbors, and finding loop closures at a later stage (§3.3).

3.1 Learning and Inference on Image Pairs

We seek a probabilistic estimate of the camera motion between a pair of images. To that end, we use

Regression Random Forests (RRF) [46], in turn based on Random Forests (RF) [47], [48]. Other supervised learning algorithms could have been used, but RRFs produce inherently probabilistic multivariate output making them an excellent fit. Testing on unseen data produces a distribution of predictions, one from each tree. We fit a Gaussian to these predictions to obtain a parametric distribution, but in principle, the raw distribution could be used. As well as their probabilistic nature, RRFs train and test quickly, can handle high dimensional feature vectors, and are trivially parallelizeable. RF algorithms have been successfully applied in a range of applications, including human pose recognition [49] and supervised mesh segmentation [50]. Relative interframe motion is modeled here using the 2D Euclidean transform, so whether training or testing, the label-space consists of three degrees of freedom: two for translation and one for rotation. In practice, we build an RRF for translation and an essentially identical RRF for rotation to reduce the amount of training data needed. The rotational RRF is trained to predict small camera rotations around the optical axis and is used for postprocessing. Differences between the two RRFs are highlighted in §3.3 and §4.

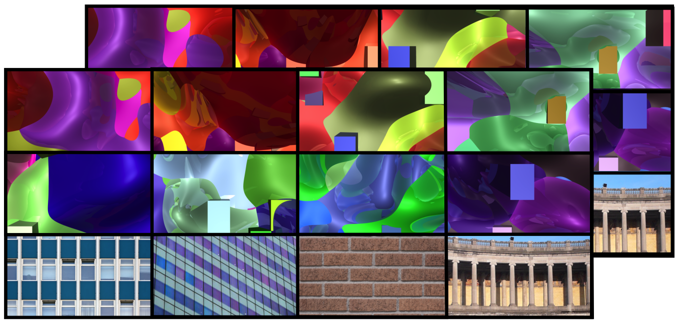


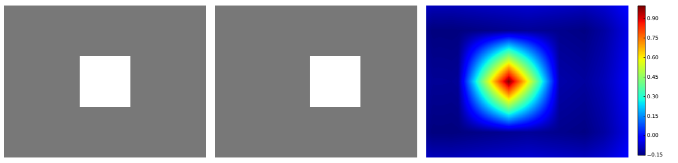
Fig. 3: The RRF is trained on thousands of two-frame image sequences, with known camera transformations. To obtain sufficient quantity and variety of camera moves and scenes, we generated the training data using a custom-built but simple graphics engine. The top 2 rows show a few examples of the procedurally generated scenes with depth variations and dynamic scene elements. The bottom row contains real images from Flickr that were mapped onto flat but moving surfaces to generate training data with repeated textures.

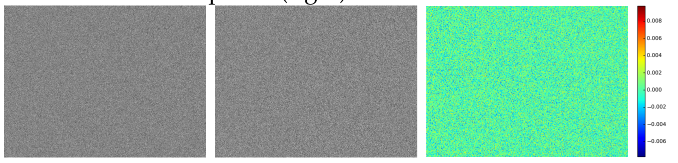
3.1.1 Training Data Acquisition

Capturing real-world video data with ground-truth camera motion is error-prone and time consuming even with specialized equipment. After a variety of attempts, including using multi-camera rigs and improved outdoor motion capture, we eventually chose to generate synthetic image pairs with known camera motion. The RRFs are able to learn how different 2D translations and rotations appear when the world is shiny, smooth, bumpy, repetitive, and when distracting objects are moving about. We did not render motion blur, but this is certainly possible. Synthesizing training data with graphics techniques has previously proved successful [14], [49], despite the obvious criticism that the resulting regressor or classifier may only be accurate on artificial-looking scenes. Aiming for large variations in shape and appearance, we rendered a family of random landscapes consisting of both angular pillars and smooth NURBS surfaces, with shape variability generated by randomly moving the pillars and deforming the surfaces. Appearance variability was achieved by rendering each object with a random color and reflectivity. We render two images of each landscape, with a random in-plane camera translation as the only difference between them. The generated images include both texture rich and texture poor regions, and irregular curved edges between NURBS surfaces, which are elusive to many interest-point detectors.

A benefit of our supervised learning approach is that if deficiencies are found in the future, it is possible to augment the training set and improve model performance. During development of our system, it was determined that regularly repeating structures posed difficulties for the system. We augmented the training set by adding "billboard" datasets which replaced the random landscape previously described with a textured polygon, containing one of a set of images of repeated structure which were obtained from Flickr and other Creative Commons sources. Example frame pairs from our training data are shown in Fig. 3.

3.1.2 Feature Computation


(a) Image pair (left/middle) with strong texture, producing unimodal NCC response (right).


(b) Textureless image pair (left/middle) producing flat NCC response (right).

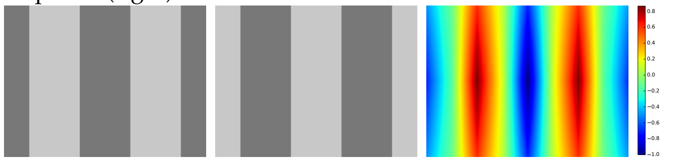

(c) Image pair (left/middle) with repeated structure producing multimodal NCC response (right).

Fig. 4: Representative types of image pair we may see (left, middle), along with their corresponding NCC response (right).

A 3599 dimensional feature vector is extracted from each image pair by encoding the responses of many Normalized Cross Correlation (NCC) comparisons between different segments of the images. NCC was chosen because it concisely describes many properties of image pairs. Images containing similar, distinctive, localizable content produce unimodal NCC responses (Fig. 4a). Textureless or uniform input images produce approximately flat NCC responses (Fig. 4b). Images with repeated structure produce periodic NCC responses (Fig. 4c). Our feature extraction aims to detect and encode all these situations, allowing the RRF to learn the mapping between NCC responses and camera movement.

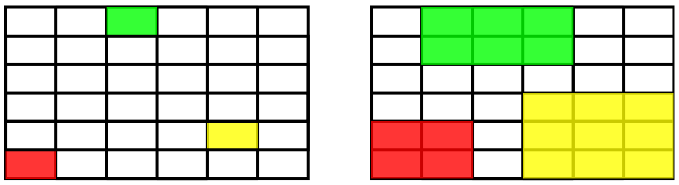


Fig. 5: template (left) and search (right) patches representing some combinations of image regions on which NCC is run (6×6 grid level). Color reflects correspondence. Note the edge truncation behavior.

For image index pair $(j, k) \in \mathcal{P}$ we take $\boldsymbol{I}_j$ as the template image and $\boldsymbol{I}_k$ as the search image. A pyramid of patches is defined by placing regular $1 \times 1, 2 \times 2, 4 \times 4, 6 \times 6$ and 8×8 grids onto each image (the 6×6 case is shown in Fig. 5). This approach of taking patches at different scales and areas of the Each patch at each grid resolution in the template image is compared using NCC to a region of the search image creating a response image $\boldsymbol{N}$. To allow for scene movement between the images, each template patch is compared to a larger region in the search image, by expanding out 1 patch in each direction unless the edge of the image prevents this. In Fig. 5 the colored patches indicate representative examples of regions that would be compared. This first step (FEATEXTRACTIMG in Fig. 6) produces 121 different NCC responses, each of which is then encoded to a few numbers, the concatenation of which forms our full feature vector.

A strongly peaked NCC response indicates a likely offset for the scene content (*i.e.* this portion of the scene contains localizable texture). In this case providing the location of this offset to our machine learning system is crucial, as (for example) if every NCC comparison contained a strongly peaked offset to the left, this is strong evidence that the camera has moved to the right. If the response is relatively flat, *i.e.* the peak value is close to the mean value, then the patches compared are likely textureless and so no definitive decision can be made about the camera motion. Note that this absence of certainty is an equally important input to our machine learning technique (it may make the RRF more likely to output a large variance). The NCC image containing a "ridge" (peak only localizable in 1 dimension) indicates certain types of scene geometry (and certain degrees of belief about possible motion) and so must also be concisely encoded. Each NCC response is encoded as follows (see ENCODENCC in Fig. 6), and the concatenation of all these defines our feature vector.

```
procedure FEATEXTRACTIMG($\boldsymbol{I}_s, \boldsymbol{I}_t$)
    for $l \in \{1, 2, 4, 6, 8\}$ do
        for $x \in \{0 \ldots l-1\}$ do
            for $y \in \{0 \ldots l-1\}$ do
                $\boldsymbol{x}, \boldsymbol{y} \leftarrow$ TEMPLPIX($l, x, y, \boldsymbol{I}_t.shape$)
                $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}} \leftarrow$ SEARCHPIX($l, x, y, \boldsymbol{I}_s.shape$)
                $\boldsymbol{T} \leftarrow \boldsymbol{I}_t[\boldsymbol{y}, \boldsymbol{x}]$
                $\boldsymbol{S} \leftarrow \boldsymbol{I}_s[\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}]$
                $\boldsymbol{N} \leftarrow$NCC($\boldsymbol{S}, \boldsymbol{T}$)
                ENCODENCC($\boldsymbol{N}, l, \boldsymbol{S}.shape$)

procedure ENCODENCC($\boldsymbol{N}, l, shape$)
    OUTPUT(MIN($\boldsymbol{N}$),MAX($\boldsymbol{N}$),MEAN($\boldsymbol{N}$))
    $x, y \leftarrow$ PEAKCOORDS($\boldsymbol{N}$)
    $x', y' \leftarrow$ NORMPEAK($x, y, shape$)
    OUTPUT($x', y'$)
    for $p \in 10, 20$ do
        OUTPUT(LAPLACECOORDS($\boldsymbol{N}, x, y, p$))
    OUTPUT(NORMEDHIST($\boldsymbol{N}, (-1, 1), 5$))
    if $l <= 2$ then
        for $\boldsymbol{G} \in \mathcal{G}$ do
            $\boldsymbol{H} \leftarrow \boldsymbol{N} * \boldsymbol{G}$
            $a, b \leftarrow$ MIN($\boldsymbol{H}$),MAX($\boldsymbol{H}$)
            $c, d \leftarrow$ MEAN($\boldsymbol{H}$),MEDIAN($\boldsymbol{H}$)
            OUTPUT($a, b, c, d$)
```

Fig. 6: Pseudocode to extract a feature vector from equal sized grayscale images $\boldsymbol{I}_s$ and $\boldsymbol{I}_t$. Indentation denotes structure, as with Python. Zero based indexing is used. $\boldsymbol{I}[\boldsymbol{y}, \boldsymbol{x}]$ is a slicing operation to extract the subwindow defined by pixel index vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. TEMPLPIX returns the pixel indices to extract a small "template" window for a given level, window index and image size (Fig. 5 left). SEARCHPIX operates similarly but produces indices for a larger "search" window (Fig. 5 right). OUTPUT appends a variable number of features to the feature vector being built for this image pair (for clarity, there is no variable to represent the feature vector in the pseudocode). PEAKCOORDS computes the 2D pixel shift for the peak of the NCC response. NORMPEAK normalises this in relation to the template image size. LAPLACECOORDS computes a Laplacian coordinate descriptor on $\boldsymbol{N}$ around the peak point (x, y) at a scale p. NORMEDHIST returns a 5 bin histogram of the NCC image with bin limits $(-1, 1)$.

Minimum, maximum and mean values are computed to give an idea of the response distribution, particularly how the peak value compares to the rest of the values. The 2D offset of the peak location is found, and the sizes of the input patches is used to convert it to a normalised offset, so that when the input patches are exactly the same and the peak indicates this, the offset $(0, 0)$ will be added to the feature vector. The shape of the response is captured by computing Laplacian Coordinates around the peak point. We apply the Laplacian operator $\frac{1}{4}\begin{bmatrix}1 & -2 & 1\end{bmatrix}$ to 1D "slices" through the 2D surface of $\boldsymbol{N}$. These slices all coincide with the peak, and are made at 4 different orientations and two different scales (so the points away from the peak which are multiplied by 1 are either 10 or 20 pixels away). This 8D descriptor encodes the shape of the peak - if all the numbers are large then the peak is strongly localizable in all directions. If all the numbers are close to zero this means the peak is very wide, and if (for example) the numbers for horizontal "slices" are low and the numbers for vertical slices are large, the peak is a horizontal "ridge". A normalised histogram with 5 equally spaced bins between -1 and 1 is also stored.

The final step, which we only carry out for the 1×1 and 2×2 grid resolutions, is to run a Gabor filter bank $\mathcal{G}$ over the image. The filters $\boldsymbol{G} \in \mathcal{G}$ vary in orientation, scale and frequency in order to try to capture the multimodal NCC response produced by images with repeating structure. The min, max, mean and median of each Gabor response is stored, thus completing our feature encoding scheme. The parameters used to generate the filter bank can be found in the supplemental material.

3.2 Layout to Global Coordinates

The pairwise estimates provided by the Random Forest use a relative coordinate system, but to navigate images as a Swipe Mosaic we require image locations in a global coordinate system. By limiting motion to a 2D plane and approximating the RRF output as Gaussian, we solve this problem in closed form using linear least squares in a similar spirit to [51]. For each $(j,k) \in \mathcal{P}$, the RRF gives a mean $\boldsymbol{\mu}_{jk} = [\mu^x_{jk} \quad \mu^y_{jk}]^T$ and standard deviation $\boldsymbol{\sigma}_{jk} = [\sigma^x_{jk} \quad \sigma^y_{jk}]^T$. An error function

$$E(\boldsymbol{x}) = \sum_{j,k\in\mathcal{P}} \left(\frac{(x_k - x_j) - \mu^x_{jk}}{\sigma^x_{jk}}\right)^2 + \left(\frac{(y_k - y_j) - \mu^y_{jk}}{\sigma^y_{jk}}\right)^2 \tag{1}$$

is defined on the vector of all camera locations

$$\boldsymbol{x} = \begin{bmatrix} x_1 & y_1 & x_2 & y_2 & \dots & x_N & y_N \end{bmatrix} \tag{2}$$

by summing squared differences between the *actual* pairwise offsets in $\boldsymbol{x}$ and the *predictions*, weighted by the prediction uncertainty. $E(\boldsymbol{x})$ can be written as $e(\boldsymbol{x})^T e(\boldsymbol{x})$ where

$$e(\boldsymbol{x}) = \begin{bmatrix} \frac{(x_{k_1} - x_{j_1}) - \mu^x_{j_1k_1}}{\sigma^x_{j_1k_1}} & \frac{(y_{k_1} - y_{j_1}) - \mu^y_{j_1k_1}}{\sigma^y_{j_1k_1}} & \dots \end{bmatrix}^T . \tag{3}$$

$e(\boldsymbol{x})$ can be written as $\boldsymbol{Ax} - \boldsymbol{b}$ where each row of $\boldsymbol{A}$ contains zeros in all but two entries, at locations to match the variables in $\boldsymbol{x}$, containing positive and negative inverse standard deviation, and the corresponding element of $\boldsymbol{b}$ is the mean prediction from the forest divided by the standard deviation. Two more rows are added to $\boldsymbol{A}$ and $\boldsymbol{b}$ to overdetermine the system by forcing (x_1, y_1) to be $(0,0)$. The unique solution for $\boldsymbol{x}$ is determined by the standard least squares method for solving $\boldsymbol{Ax} = \boldsymbol{b}$.

3.3 Post-processing

Translational Loop Closure

Even with good predictions, errors accumulate over long image sequences, meaning camera path loops may not line up correctly in the output. To mitigate this problem, from the initial layout we automatically find "loop points" - pairs of images which are close in the 2D coordinate space but temporally distant. We compute each image's five nearest spatial neighbors, and any neighbors whose timestamps differ by $> n$ become loop points. For our sequences we set n at 25. The number of pairs chosen with this technique is scene-dependent, but it is typically orders of magnitude less than the number of temporal pairs used to make the initial layout. From each loop point image pair, we compute a feature vector and corresponding prediction from the translational RRF. A new layout is computed from the combined set of temporal and loop-based predictions.

Rotational Correction

The majority of video sequences will contain small variations in rotation about the optical axis, which may be difficult to see when viewing frames in order. When our viewer (§3.4) transitions across loop points containing this kind of rotation, even a few degrees disparity is enough to produce a noticeable artifact. The visualization can render the images with rotation correction, but needs to be provided with the amount to rotate each image by. A second "rotational" RRF was built using the same feature vector as before, but trained to predict optical axis rotation between two images. Corresponding synthetic training data was rendered using the same method as for translation, with the only difference being the camera undergoing a random rotation around the optical axis instead of a random translation.

Because our training data for this RRF contains image pairs *only* affected by rotation, performance was poor on images containing rotation *and* translation. To avoid this, we crop each image such that the centers of the cropped images should contain the same scene point, and thus the cropped images differ only in camera rotation. Our 2D translation prediction from the layout algorithm is used to calculate how much to crop the images by. As with the translational loop closure, we predict rotations for image pairs which are found at "loop points". For each image pair, features are generated from the cropped images, and the rotational RRF returns a 1D probabilistic estimate. Relative rotation estimates are combined using an analagous technique to the least squares layout algorithm (§3.2). Linear constraints encourage the relative rotational difference between two frames to match the predictions, encourage the rotations to be close to zero (this is a hard constraint for the first image only), and encourage smoothness between temporal neighbors.

3.4 Swipe Mosaic Interface

We have implemented our viewer interface as both a native desktop application, using Python and OpenGL, and a web app using JavaScript and WebGL, allowing users to browse Swipe Mosaics without installing any software. Whichever interface is used, Swipe Mosaics are viewed by first loading in images and camera locations. The interface shows a single image in sharp focus at any one time, and optionally shows blurred surrounding images. The user navigates by clicking anywhere on the image and dragging ("swiping"). The swipe direction determines where on the 2D map the program looks for a new frame to transition to. If the user swipes enough to the left, we transition to rendering the neighboring frame on the right, in a manner similar to PDF viewers and services such as Google Maps. An "minimap" in one corner of the interface conveys an idea of the overall layout of the scene and highlights the location of the currently displayed image. As an alternative navigational aid, users can enter "Picasso view" which smoothly zooms out and displays all the images overlaid. These images are not intended to line up perfectly as in a panoramic mosaic, but rather to provide a sense of which directions can be navigated. While the user presses down and swipes to navigate to a new image, we render smooth transitions using alpha blended crossfades. The on-screen positions of the current and next frames move smoothly in sync with the mouse, with the intention that if the user clicks on a particular recognizable feature in the scene, that feature will remain close to the cursor no matter where it is moved. If a rotation vector (see §3.3) is provided to the viewer, images are rendered with the corresponding rotational correction.

4 Experiments

Experiments were performed on videos filmed by our users on mobile phones, camcorders and SLRs, along with videos from other sources which were not captured with this purpose in mind. We examine the performance of the pairwise regressor, and the overall Swipe Mosaic rendering and visualization system quality. We perform qualitative and quantitative comparisons to validate our system's ability to handle a variety of sequences, which are listed along with their defining characteristics in Table 1. A number of baseline algorithms are compared to, which represent different approaches to this task in the existing literature.

To reiterate, there are many methods that produce camera paths on simple, textured, static, planar scenes. When they work, they too could be used to prepare a sequence for use as a Swipe Mosaic. We compare to specific representative baseline methods to demonstrate that our approaches degrades gracefully with footage that is less simple, in a varity of ways.

Sequence	Characteristics
FENCE∗	Easy sequence, abundant texture
MINI	Abundant texture
LOBBY∗	Abundant texture, demonstrates loop closure
FACADE	Abundant texture
GRATING	Partially textureless
RAILS	Large scale repeated structure
SKATER	Dynamic and deforming foreground object
FLOWERS†	Dynamic objects, non planar motion, [52] failure case
SCULPTURE	Little texture, motion blur
LEAVES	Dynamic and deforming scene
OBELISK	Non planar camera path
HANDBAG	Dynamic specularities
WALL	Ambiguous motion due to repeated structure
VINYL	Motion blur, textureless occluding object
ISS∗	Demonstrates Loop closure
DINO	Moving background elements
PRISM	Automatic gain control, CCD overload
AQUATIC†	Scene from movie, contains dynamic objects + parallax
FREIBURG2†	6D Ground truth available

TABLE 1: Test sequences along with their defining characteristics. ∗ - only appears in the supplemental material. † - captured without intended purpose of building a Swipe Mosaic.

4.1 Regressing Motion Between Image Pairs

We start by inspecting what the regressor has learned about the relationship between the computed features and estimating translations. When images contain unambiguous texture (Fig. 7a) our regressor is confident in both x and y. For scenes with repeated texture but a unique vertical structure (Fig. 7b), the regressor outputs small σ^x and large σ^y, reflecting the uncertainty caused by the aperture problem. Fig. 7c show the result of using the same type of regressor and features, but training to estimate the in-plane rotation between two images. We expect the rotational RRF to perform better with a customised feature vector, but the vector designed for translation allowed rotational correction within $\pm 5°$.

Initial versions of our feature extraction used Optical Flow (rather than NCC) on the input images, before condensing that information into a vector. Building the feature vector from NCC is not an obviously better choice than using Optical Flow, but we obtained better test-time results with NCC based features. It is likely that the NCC responses are better correlated with motion-confidence than flow, which has some estimated vector for every pixel. Significantly, the regressor has the benefit of learning from our graphics engine: it has seen thousands of rendered examples of image pairs, with knowledge of the true 2D Euclidean transform.

The RRF training process selects some elements of the feature vector more frequently than others for estimating the transform parameters at test time. Fig. 8 shows spatial histograms for each scale of the NCC grid, indicating the frequency with which a feature computed from that NCC sub-window was chosen by the forest training process. Interestingly, the most used level is the 4×4 grid, and there is a strong peak within that histogram for the most central 4 of the 16 possible NCC sub-windows. While using a single NCC calculation to compare whole images is a common approach for image alignment, these graphs show that the finer grained NCC sub-windows are providing important extra information to get the right offset. The top level of features (representing a single global NCC comparison) are chosen by the forest training process 246 times, whilst the 4×4 resolution features are chosen 665 times. We know that this is due to these features being more informative, rather than simply more numerous, because the 8×8 level, which contains a larger number features than all the other levels

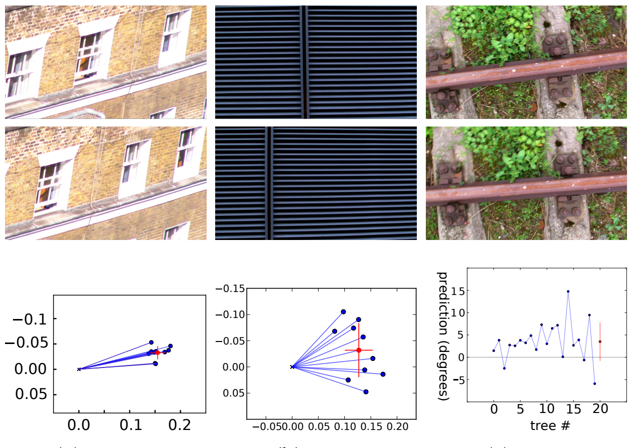

(a) FACADE (b) WALL (c) RAILS

Fig. 7: In each column the top two images were inputs resulting in the RRF output at the bottom. The graphs in a) and b) show 2D results from the Translational RRF, and c) shows 1D results from the Rotational RRF. Blue dots show individual tree outputs, red dots & bars show the mean & variance of the fitted Gaussian.

put together, is only chosen 121 times.

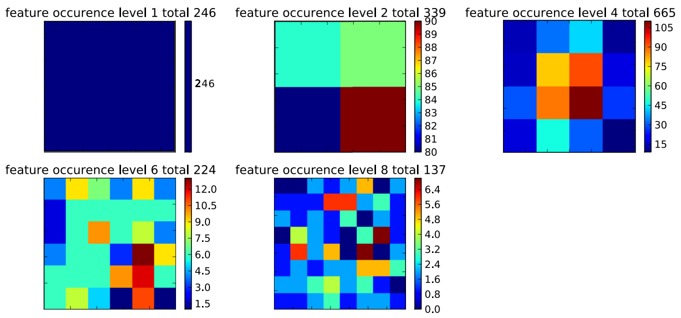


Fig. 8: Spatial histograms showing the quantity and arrangement of features chosen most frequently during training from each level of the NCC grid, for our translational RRF.

4.2 Swipe Mosaic Visualization

A key property of Swipe Mosaic visualization is being able to grab elements and navigate spatially between temporally distant frames. As shown in the video, it is possible to easily maintain a sense of position while navigating within the wider scene. Possible applications for the system include recording the damage in a car accident for later scrutiny, or examining products when shopping online.

4.3 Suitable Image Sequences

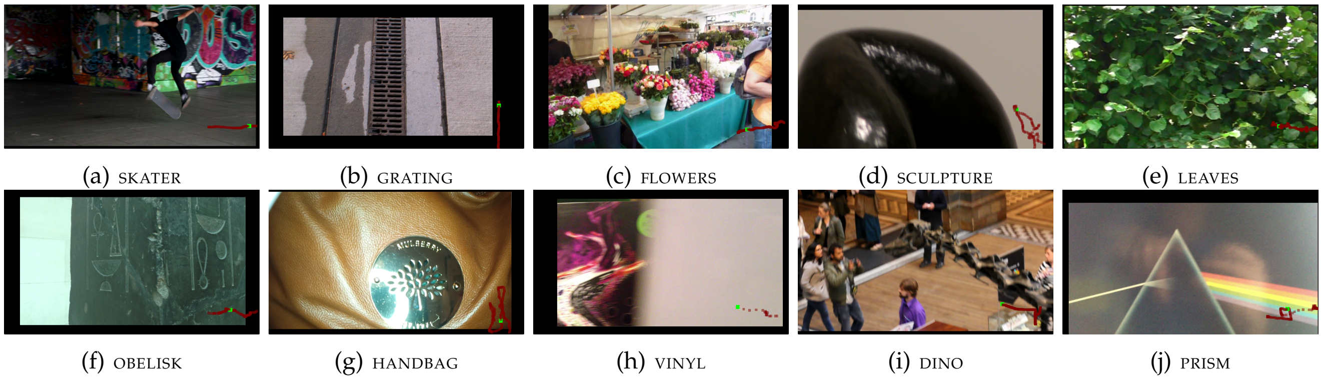
Swipe mosaics are best observed in motion, so we present qualitative results in the video. Our system performs best when the camera motion and scene geometry are parallel and both approximately planar (as with the training data), but we are robust to deviations from this setup. OBELISK shows that if the object of interest fills much of the screen, we infer a 2D version of the motion as the camera *rotates around* the object. DINO contains people in the background moving in various directions, but the transforms computed allow navigation along the main item of interest. The level 4 histogram in Fig. 8 helps explain this; the RRF learns that the center of the image is usually more informative, and so treats the motion implied by central pixels as more informative than that implied by edge pixels. View dependent effects such as the specularities in the HANDBAG sequence do not lead to incorrect motion estimates. SKATER, however, features non rigid movement in the *centre* of the frame and only the *outskirts* imply the (correct) sideways motion.

To demonstrate the utility of our system on existing sequences filmed by others, we processed a scene from the movie “The Life Aquatic with Steve Zissou”. AQUATIC features the camera panning over a cutaway version of a boat, travelling between rooms and showing different characters. The camera trajectory roughly matches the assumptions made in our training data, but the scene contains large amounts of parallax due to depth variation, as well as dynamic characters. We put 600 frames into our system and built a swipe mosaic which allows intuitive navigation between seven distinct areas. Sample frames from transitioning “through” a wall are shown in Fig. 10. Please see the supplemental video to view this scene in motion.

Sequences such as HANDBAG (Fig. 9g) can be processed successfully despite containing out-of-plane camera translation and strong specularities. Note that if a loop point featured images with differing scale, this would pose a problem for our system both in terms of the loop closure algorithm and in terms of viewer artifacts. To test the limits of our system, we ran it on a challenging part of the FLOWERS scene, a failure case from [52], which the authors describe as troublesome due to the pedestrians occluding geometry and cutting feature trajectories. The camera is moving forwards as well as sideways so that the contents of the flower stall appear to be moving roughly diagonally in image space. Our system copes with this motion, and we can browse the scene by swiping elements on the flower stall. This challenging video also demonstrates a failure mode of our system; obstructing bystanders in the image centre, combined with motion that differs significantly from that of our training data, makes for a very difficult scene.

4.4 Qualitative Evaluation against Baseline Algorithms

We evaluate the performance of our odometry regressor by comparing to simple NCC based alignment [25], VisualSfM by Wu *et al.* [53], [54], Viewfinder Alignment by Adams *et al.* [55], Real-time image-based tracking of planes using Efficient Second-order Minimization (ESM) by Benhimane & Malis [56], and “microSfM” or “μSfM”, a new system which uses the methodology of Fundamental Matrix computation but produces a 2D translation. While these techniques can be applied to a wide range of sequences, we confirmed known circumstances under which each of the baselines failed, and



(a) SKATER (b) GRATING (c) FLOWERS (d) SCULPTURE (e) LEAVES (f) OBELISK (g) HANDBAG (h) VINYL (i) DINO (j) PRISM

Fig. 9: Screenshots of various sequences loaded in the Swipe Mosaic viewer

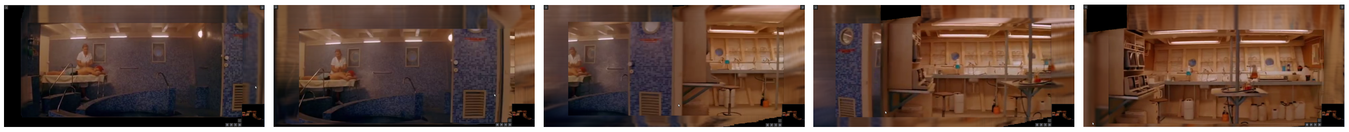


Fig. 10: A series of screenshots taken during a single swipe, navigating AQUATIC, a scene generated from the movie "The Life Aquatic with Steve Zissou"

our technique succeeded. Our baseline comparisons are summarised below; please see the Appendix and video for details.

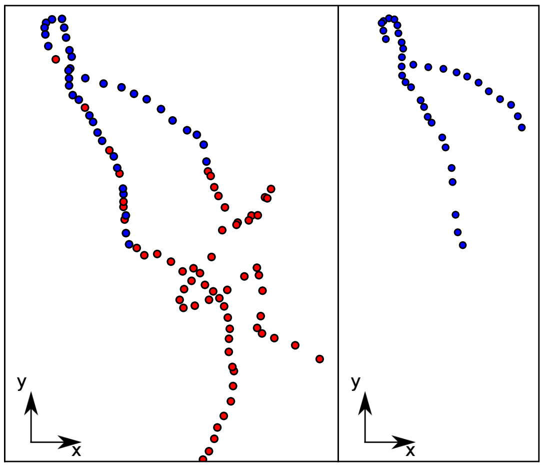
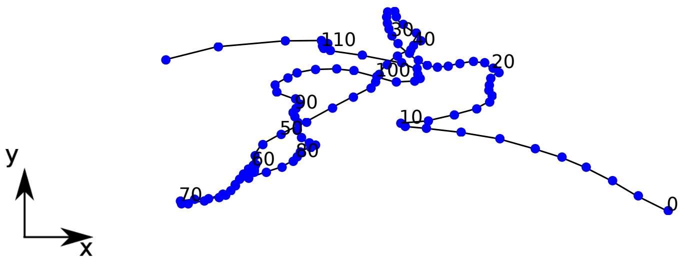


Fig. 11: 2D camera coordinates (unitless) for SCULPTURE produced by our system (left) and VisualSFM (right). Blue points indicate images where both systems gave an estimate of location (note the estimates differ); red points are images where only our system produced an estimate. NB: One outlier is not shown in the right hand image

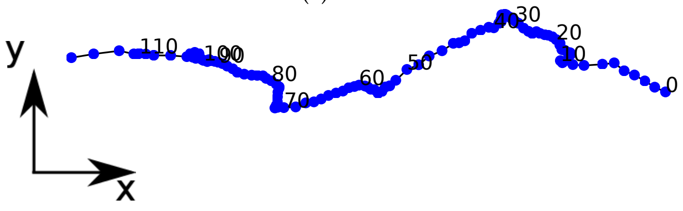
Our NCC based feature encoding is partly inspired by the **Direct Methods** detailed by Szeliski [25], which compute the best 2D alignment between images from the location of the peak in their combined NCC response. This simple method often succeeds, but we found sequences where the NCC method produced incorrect results. Szeliski [25] notes *"[NCC's] performance degrades for noisy low-contrast regions"*. For example, in the PRISM sequence where horizontal movement takes place, the failure occurs on two specific frames where the resulting NCC image had a maximum implying a translation of $(0, 0)$. The NCC response contained a second mode, corresponding to a more sensible horizontal offset, but the height of this "correct" peak was slightly lower than the peak representing zero motion, so it would never be chosen by the alignment algorithm. Systems deterministically selecting the global NCC peak and ignoring other factors will fail on this scene and similar scenes. Our RRF incorporates this peak offset information as part of the feature vector.

VisualSfM [53], [54] is a state of the art SfM system, which can process images either as "ordered" (temporally sequential frames) or "unordered". It produces excellent results in general, but struggles when few or misleading feature matches are present. For example, we evaluate on two sequences: SCULPTURE containing motion blur and textureless regions, and LEAVES containing moving geometry. VisualSfM failed to return a full, correct result for either scene, whereas in both cases our system produced a full layout suitable for browsing as a Swipe Mosaic. For SCULPTURE, the best result was with ordered mode, but only 38 camera positions were returned for an input of 101 images (Fig. 11). Camera positions were produced for all 201 frames in LEAVES using unordered mode, but the location accuracy becomes progressively worse throughout the sequence (see Video).

μ**SfM** is a system of our own creation using traditional SIFT matching and RANSAC to compute a 2DoF translational offset rather than a 7DoF fundamental matrix. Though designed to be robust in many scenarios, The VINYL sequence, containing a textureless obstruction close to the camera, causes μSfM to fail. Descriptors computed on the obstruction display self-similarity, resulting in noisy matches. Frames containing the obstruction were laid out far to the right of the rest of the scene, when they should be in the middle. Our system produces a correct horizontal motion path. A similar failure occurs in PRISM, which features a typical home video problem



(a) ESM



(b) Ours

Fig. 12: 2D camera coordinates produced for OBELISK using ESM (Fig. 12a) and our method (Fig. 12b). The true camera motion is approximately constant horizontal translation, coupled with rotation to remain pointed towards the object of interest (see supplemental video).

of the camera moving into direct sunlight, and the automatic gain control taking a few frames to adapt. No interest points were matched between the beginning and end of the sequence, so camera locations cannot be inferred for the whole sequence. Our system produces a zero mean, high variance estimate (effectively applying a Brownian Motion prior) whenever there is no visual evidence suggesting any particular direction of motion, allowing the resulting (complete) set of camera locations to be browsed as a Swipe Mosaic.

Real-time image-based tracking of planes using Efficient Second-order Minimization (ESM) [56] is a direct method which explicitly models the scene as a plane, and searches for a parameterised transform which minimises the sum of squared differences between two images. The transform can be parameterised as anything from a full homography (8DoF) to a translational transform (2DoF), and the parameters are solved for using an efficient method which achieves Newton method like convergence rates, without having to compute the Hessian. We used the implementation of ESM available in Ed Rosten's LibCVD project, using 2 DoF to produce a translation between each image pair, before using our layout algorithm. ESM produces a good Swipe Mosaic result for some of our test sequences, but the explicit parameterisation of the scene as a plane leads to problems when faced with non-planar camera paths or distorting objects. ESM performed very badly on the OBELISK scene, laying out frames which should be very far from each other in roughly the same place. Our system produced an intuitively navigable Swipe Mosaic. The camera paths produced by ESM and our method are shown in Fig. 12. Note the broadly horizontal linear path produced by our method in contrast to the ESM path which continually crosses itself. This is due to our method's more gradual degradation as scenes deviate from planar, allowing us to cope with strong perspective deformations. See the supplemental video for a comparison of the browsing experience of these two solutions.

Viewfinder Alignment (VfA) [55] computes constrained transforms between temporally close video frames. A "digest" containing edge information in multiple orientations, and the locations of the strongest detected corners is computed for each frame. Two digests are matched by aligning the histograms to give a putative 2D shift, and aligning the detected corner locations to evaluate how likely this shift is to be correct. VfA works well given sufficient strong edges and recognisable corners, but if either are absent it will produce an incorrect transform, or no transform at all. The GRATING sequence contains similar strong edge information in all frames, leading VfA to erroneously believe geographically distant frames were close together. Apart from the edges, sufficient visual cues are present in the image so that both our system and μSfM inferred a correct result. Because only the location, and no visual descriptor, is stored for each corner, VfA is prone to linking disparate frames with similar edges. Low texture regions in PRISM or VINYL cause no corners to be detected, meaning VfA cannot compute a result. Further results are included in the supplemental material.

4.5 Quantitative Evaluation Against NCC

While graceful degradation is easy to illustrate qualitatively, it is reasonable to check if regression using our NCC-based feature vectors is actually different than just using NCC directly, at least for best-case in-plane motion and static scenes with negligible motion blur. To quantitatively evaluate the odometry of our system, we compare on the FREIBURG2 sequence from the TUM [57] dataset. This dataset consists of Kinect video sequences alongisde 6D ground truth camera positions, generated using 100Hz active motion capture. Unlike the rest of the dataset, the camera path in FREIBURG2's first 950 frames contains almost exclusively vertical and horizontal translation. This is the kind of motion that both NCC and our system should be able to handle. The appearance of the indoor office scene is unremarkable in terms of texture or dynamic elements.

Both NCC and our system were used to generate 2D camera locations for the sequence. Every pair within a 4 frame temporal window was used to generate relative offset predictions. For both systems, the offsets were fed to our least squares layout algorithm. To compare the 6D ground truth poses with each 2D solution, three steps had to be carried out. First, a ground truth pose must be established for each RGB frame, as the dataset only provides camera locations from motion capture, and the Kinect and motion capture systems were running unsynchronised at different frequencies. Secondly, some projection of each 6D camera pose onto 2D must be established, and finally the 2D solutions must be scaled and aligned to assess the correctness of the locations.

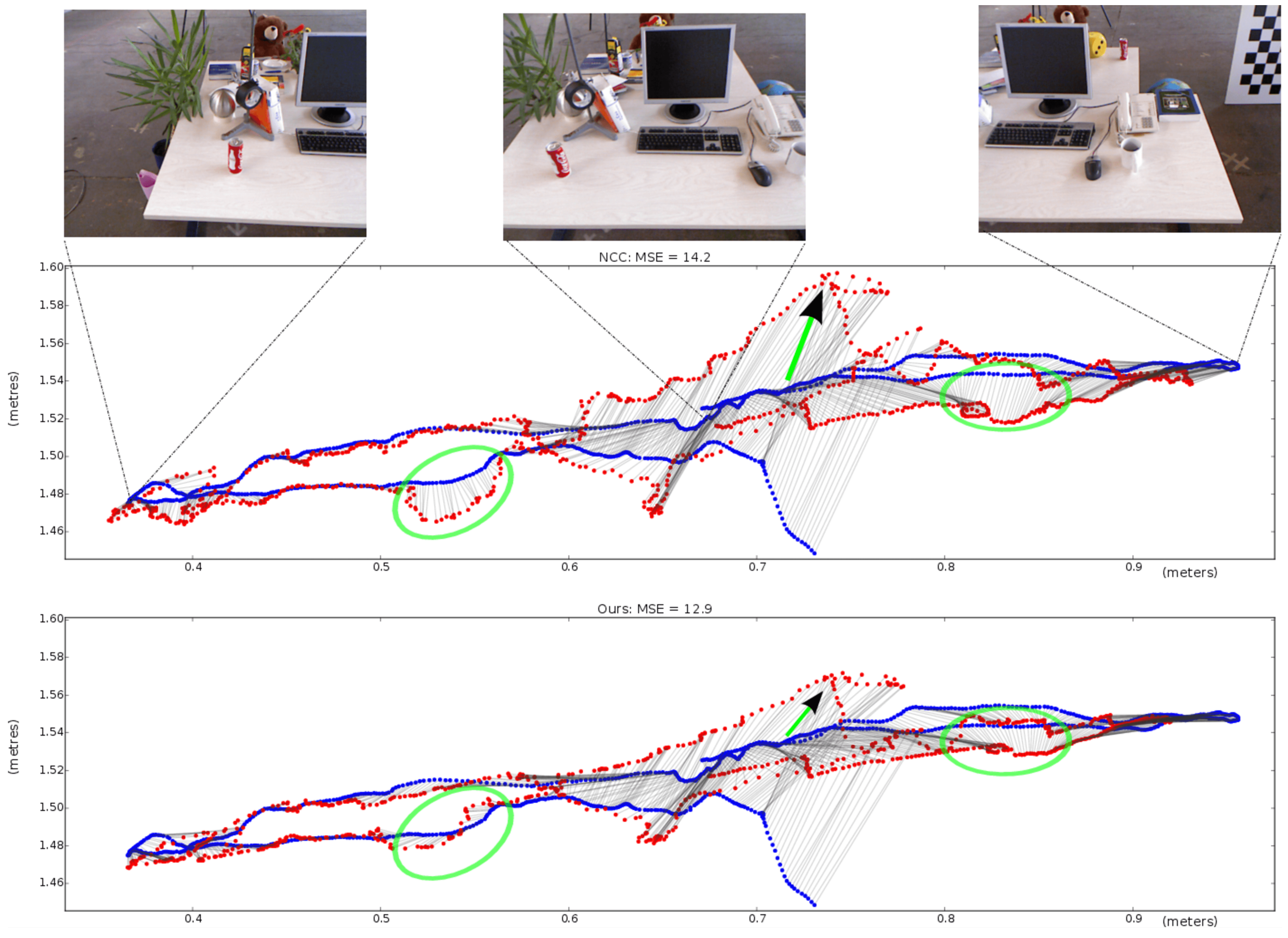


Fig. 13: Alignment result for NCC (top) and our system (bottom) for the FREIBURG2 sequence [57]. Each graph shows the camera locations from the prospective solution (Red) after being aligned using Procrustes to the ground truth locations computed with a "best fit" plane (Blue). Corresponding camera locations are joined by grey lines. Note the highlighted regions (green ellipses / arrows) in which our system provides a solution substantially closer to ground truth. Navigating these areas of the NCC solution as a Swipe Mosaic would require users to follow a more distorted trajectory than would seem natural. Corresponding frames are shown above to give an idea of the scene makeup. Onscreen viewing recommended.

To establish a 6D ground truth location for each RGB frame, we linearly interpolated between the closest two motion capture positions using the globally synched timestamps (available for both motion capture and Kinect readings). The rotation was encoded as a quaternion during this process, ensuring linear interpolation is a reasonable approach. The result of this operation is a 6D camera pose corresponding exactly to each RGB frame. The 6D to 2D projection step is described below, and the final alignment step is carried out using the Procrustes algorithm [58].

Given a 3D plane represented as a unit normal $\hat{\boldsymbol{n}}$ and a point on the plane $\boldsymbol{p}$, we define two unit vectors $\hat{\boldsymbol{u}}_1$ and $\hat{\boldsymbol{u}}_2$ such that $\hat{\boldsymbol{u}}_1^T\hat{\boldsymbol{u}}_2 = \hat{\boldsymbol{u}}_1^T\hat{\boldsymbol{n}} = \hat{\boldsymbol{u}}_2^T\hat{\boldsymbol{n}} = 0$. The exact orientation of these vectors is not important as the subsequent alignment includes a rotation step. Each camera location $\boldsymbol{c}_i$ is projected onto a 2D location on the plane $(\hat{\boldsymbol{u}}_1^T\boldsymbol{c}_i, \hat{\boldsymbol{u}}_2^T\boldsymbol{c}_i)$ (see Fig. 15). To define the plane orientation, we tried both a "best fit" to all the camera locations, and also tried using the "up" and "right" vectors of each individual camera in turn. The "best fit" plane was defined by setting $\hat{\boldsymbol{n}}$ to the average forward direction of each camera. The maximum angular difference between the computed normal and the forward vector of any of the cameras is 9.46 degrees, confirming that the scene contains only minimal rotation and is therefore a good candidate sequence for this comparison.

On the best fitted plane, NCC gave a final Mean Squared Error of 14.2 cm^2 against our system's 12.9 cm^2, an improvement of 9.4%. The alignments resulting from both systems are shown in Fig. 13. It is interesting to note that both algorithms make similar mistakes, owing perhaps to our system being built on top of NCC based features, but our system produces errors of smaller magnitude, especially towards the center of the horizontal axis, because it incorporates more information than just the top level, entire-image NCC comparison. As well as comparing with this fitted plane, we tried aligning to

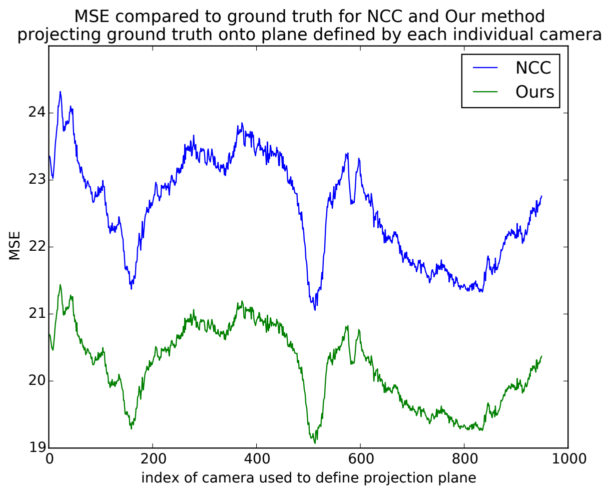


Fig. 14: Comparison of Mean Squared Errors between our method and NCC for the whole range of possible individual camera projection planes, for FREIBURG2. Note the self similarity of the lines. As we change which ground truth camera defines the projection plane, both solutions (being so similar) have coinciding increases or decreases in performance.

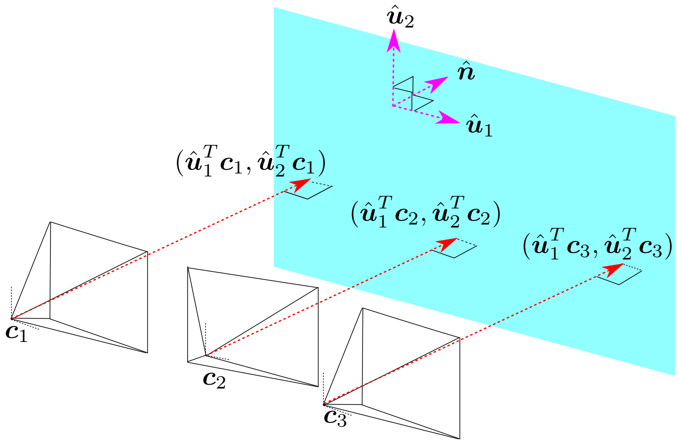

$\hat{u}_2$
$\hat{n}$
$\hat{u}_1$
$(\hat{u}_1^T c_1, \hat{u}_2^T c_1)$
$(\hat{u}_1^T c_2, \hat{u}_2^T c_2)$
$(\hat{u}_1^T c_3, \hat{u}_2^T c_3)$
c_1
c_2
c_3

Fig. 15: Diagram showing how 3D camera positions c_1, c_2, c_3 are projected (red dashed line) onto a 2D coordinate space defined by a plane (blue). For the "best fit" case, the plane normal $\hat{n}$ is generated by averaging the forward vectors of all the cameras. The basis vectors $\hat{u}_1$ and $\hat{u}_2$ lie within the plane and are mutually orthogonal. When fitting to the plane defined by an individual camera, $\hat{u}_1$ and $\hat{u}_2$ are chosen to be parallel to the "right" and "up" vectors of individual cameras (black dotted lines).

each of the planes defined by one individual camera's orientation, by projecting all other cameras onto the local "right" and "up" vectors. The results of this are shown in Fig. 14. Unsurprisingly, whichever one of the 950 cameras we choose, we see an improvement with our system, as shown by the green line always being underneath the blue line. This evaluation shows that even for a texture-rich real world sequence, not captured with Swipe Mosaics in mind, our system produces a measurable improvement over the alignment produced by NCC.

Our evaluations show the robustness of our system to visual phenomena found difficult by other systems. In the presence of dynamic objects, lack of texture, or repeated structure, we are able to compute 2D locations which enable browsing of the scene, degrading gracefully in the presence of inconclusive visual information. Many of our sequences were captured by users unfamiliar with the workings of the system, and our success here demonstrates our robustness to input data straying outside the assumptions of the training data. All results were generated by a single trained translational RRF with 10 trees and maximum depth 12. At each node, 2000 possible feature splits were considered. The one-time training took 1h 43m on a Core 2 Duo 2.8 GHz, using both CPU cores. All experiments were carried out with the same synthetic training set built from 8800 image pairs. The rotational RRF was trained with 20 trees of depth 12, considering 100 feature splits per node. Only 400 image pairs were used in the training set, as inferring pure rotation is a strictly easier problem than translation, because what appears in the images is not scene geometry dependent. All images were 768x432 or 640x480. A 2.5 GHz single core Xeon, computed 121 NCC matches and the 3599D features for two images in 15 seconds. Most useful datasets contain thousands of image pairs, so we used a cluster to process datasets quickly.

5 Conclusion

Our results show that Swipe Mosaics can be used to intuitively navigate video sequences containing a range of camera motions and visual content, including those that failed or trouble existing standard baselines. As seen in the video, even passively observing someone else's Swipe Mosaic interaction provides a good sense of a scene's layout. Traditional panoramic image mosaics undoubtedly have a cleaner overall appearance than our Picasso-view. Yet the payoff of browsing video frames through our interface is enormous: footage exhibiting parallax and other view- and lighting-dependent effects can be visualized without the extra user effort needed for most multi-perspective renderers (*e.g.* [59]), because the pixels need not join up. Training our RRFs on synthetic data has led to a visual odometry system that achieves our goals. It manages to estimate visually-acceptable translations and rotation both in textured scenes, where existing methods also work well, but also in much more difficult scenes, where other methods become brittle or fail. It is certainly possible that our model may learn still better correlations between appearance and pose if, for example, 3 or more frames were examined together, potentially allowing motion models to be incorporated. Quite significantly from the perspective of potential users, our adapted regressor-layout pairing takes account of ambiguities when computing distributions over possible camera-motions. This means that our visualization prototype fails more gracefully than existing systems that are designed for somewhat idealized conditions. In an indirect way, our RRF is looking at thousands of examples to learn its own version of the user-sought visual continuity: which parts of an image pair are correlated with each other, and how? While still challenging to dissect, the visualization of our RRF in Fig. 8 shows that NCC comparisons from particular parts of the image, at particular scales, were learned to be the most informative for this task.

5.1 Limitations and Future Work

Some kinds of "unexpected" motion (motions not featured in the training set) are handled well by our system, but others are not. When most scene geometry is moving in a similar manner, we are able to produce sensible Swipe Mosaics despite the presence of, for example, forward motion (which the RRF has not been trained on). However, as shown by the FLOWERS example, occluders dominating the image center can cause failures. Our model has learned to rely on the center of the image somewhat more than other areas, so it is likely that an enhanced feature vector and training set would be required to cope with this problem. Object-recognition could also be incorporated, *e.g.* to recognize and ignore pedestrians. Our NCC based feature vector performs well, but other features could replace or augment it. VfA's edge "digest" is appealing in this regard as it is quick to compute and could possibly be extended to describe a distribution over different alignments. Supporting more camera degrees of freedom in one RRF is desirable, but likely to require much more training data. Continuing to target individual RRFs at only 1 or 2 degrees of freedom each, as in this work, seems a promising approach. A valuable improvement to the interface would be to give live feedback at capture time about what parts of the scene require more detailed recording. The DTAM-based feedback in [38] may not be possible if scenes lack reliable interest points, but we could provide the RRF the inertial and gyroscope readings that are available in many smartphones. For now, the system is device agnostic, making it easy to create Swipe Mosaics. Swipe Mosaics will hopefully encourage content-creators to document and share the details of the world around them.

ACKNOWLEDGMENTS

The authors acknowledge the use of the UCL *Legion* High Performance Computing Facility (Legion@UCL), and associated support services, in the completion of this work. We would also like to thank the reviewers for their helpful advice, UCL staff and colleagues for their suggestions and data collection assistance throughout this project, particularly Tim Weyrich, Lourdes Agapito, Maciej Gryka, Oisin Mac Aodha, Yotam Doron, Michael Firman, Dominic Waithe, and James Durrant.

REFERENCES

[1] Photosynth, http://www.photosynth.net, 2012.
[2] D. B. Goldman, C. Gonterman, B. Curless, D. Salesin, and S. M. Seitz, "Video object annotation, navigation, and composition," in *UIST*, 2008.
[3] T. Karrer, M. Weiss, E. Lee, and J. Borchers, "Dragon: A direct manipulation interface for frame-accurate in-scene video navigation," in *CHI*, 2008.
[4] P. Dragicevic, G. Ramos, J. Bibliowitcz, D. Nowrouzezahrai, R. Balakrishnan, and K. Singh, "Video browsing by direct manipulation," in *CHI*, April 2008.
[5] C. Nguyen, Y. Niu, and F. Liu, "Direct manipulation video navigation in 3d," in *CHI*, 2013.
[6] M. Irani, P. Anandan, and S. Hsu, "Mosaic based representations of video sequences and their applications," in *ICCV*, 1995.
[7] N. Snavely, S. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," *SIGGRAPH*, 2006.
[8] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, and R. Klowsky, "Ambient point clouds for view interpolation," *TOG*, 2010.
[9] S. E. Chen and L. Williams, "View interpolation for image synthesis," *SIGGRAPH*, 1993.
[10] R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, vol. 16, pp. 22–30, 1996.
[11] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps," ser. SIGGRAPH, 1997.
[12] J. Campbell, "Techniques for evaluating optical flow for visual odometry in extreme terrain," *IROS*, 2004.
[13] S. Peleg, B. Rousso, A. Rav-Acha, and A. Zomet, "Mosaicing on adaptive manifolds," *PAMI*, 2000.
[14] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow, "Learning a confidence measure for optical flow," *PAMI*, 2012.
[15] A. Adams, E.-V. Talvala, S. H. Park, D. E. Jacobs, B. Ajdin, N. Gelfand, J. Dolson, D. Vaquero, J. Baek, M. Tico, H. P. A. Lensch, W. Matusik, K. Pulli, M. Horowitz, and M. Levoy, "The frankencamera: an experimental platform for computational photography," *TOG*, 2010.
[16] S. You and U. Neumann, "Fusion of vision and gyro tracking for robust augmented reality registration," in *VR*, 2001.
[17] G. Klein and T. Drummond, "Tightly integrated sensor fusion for robust visual tracking," *Image and Vision Computing*, 2004.
[18] J. Kopf, M. Uyttendaele, O. Deussen, and M. F. Cohen, "Capturing and viewing gigapixel images," *SIGGRAPH*, 2007.
[19] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," ser. VR, 2010.
[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.
[21] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends. Comput. Graph. Vis.*, 2007.
[22] M. Brown and D. G. Lowe, "Recognising panoramas," *ICCV*, 2003.
[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 1981.
[24] A. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision (2. ed.)*. CUP, 2006.
[25] R. Szeliski, "Image alignment and stitching: a tutorial," *Found. Trends. Comput. Graph. Vis.*, 2006.
[26] J. Davis, "Mosaics of scenes with moving objects," *CVPR*, 1998.
[27] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *AI*, 2001.
[28] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *PAMI*, 2007.
[29] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in *CVPR*, 2011.
[30] K. Murphy, Y. Weiss, and M. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 1999, pp. 467–475.
[31] R. Garg and S. M. Seitz, "Dynamic mosaics," *3DIMPVT*, 2012.
[32] A. Agarwala, K. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski, "Panoramic video textures," in *TOG*, 2005.
[33] A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg, "Dynamosaics: Video mosaics with non-chronological time," in *CVPR*, vol. 1, 2005, pp. 58–65.
[34] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *TOG*, 2011.

[35] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The lumigraph," *SIGGRAPH*, 1996.
[36] M. Levoy and P. Hanrahan, "Light field rendering," *SIGGRAPH*, 1996.
[37] H. Shum and L. He, "Rendering with concentric mosaics," in *SIGGRAPH*, 1999.
[38] A. Davis, M. Levoy, and F. Durand, "Unstructured light fields," in *Computer Graphics Forum*, vol. 31, no. 2pt1, 2012.
[39] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, 2007.
[40] S. A. Holmes, G. Klein, and D. W. Murray, "An o (n^2) square root unscented kalman filter for visual simultaneous localization and mapping," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 7, pp. 1251–1263, 2009.
[41] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," in *TOG*, 2006.
[42] J. Kopf, B. Chen, R. Szeliski, and M. Cohen, "Street slide: Browsing street level imagery," *SIGGRAPH*, 2010.
[43] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski, "Finding paths through the world's photos," *SIGGRAPH*, 2008.
[44] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in brain research*, 2006.
[45] J. Tenenbaum, V. De Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[46] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning," *Found. Trends. Comput. Graph. Vis.*, 2011.
[47] L. Breiman, "Random forests," *Machine Learning*, 2001.
[48] T. K. Ho, "Random decision forests," in *Document aAnalysis and REcognition, 1995. Proceedings of the Third International Conference on*, 1995.
[49] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-Time Human Pose Recognition in Parts from a Single Depth Image," *CVPR*, 2011.
[50] E. Kalogerakis, A. Hertzmann, and K. Singh, "Learning 3d mesh segmentation and labeling," *TOG*, 2010.
[51] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor estimates," in *ICRA*, 2006.
[52] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," *ACM Trans. Graph.*, 2012.
[53] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," http://cs.unc.edu/ ccwu/siftgpu, 2007.
[54] C. Wu, S. Agarwal, B. Curless, and S. Seitz, "Multicore bundle adjustment," *CVPR*, 2011.
[55] A. Adams, N. Gelfand, and K. Pulli, "Viewfinder alignment," *Eurographics*, 2008.
[56] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2004, pp. 943–948.
[57] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.
[58] S. Prince, *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012.
[59] A. Rav-Acha, P. Kohli, C. Rother, and A. Fitzgibbon, "Unwrap mosaics: a new representation for video editing," in *TOG*, vol. 27, no. 3, 2008.

Malcolm Reynolds received the BA (Hons) in Computer Science from Queens' College Cambridge in 2008, before moving to University College London where he was awarded the MSc Machine Learning in 2009. He is currently completing the EngD in Virtual Environments, Imaging and Visualization under the supervision of Gabriel J. Brostow at UCL. His interests include applying Machine Learning techniques such as Supervised Learning and Belief Propagation to Vision problems, particularly camera odometry, and he has previously worked on computing confidence metrics for Time-of-Flight data.

Tom S. F. Haines Tom S.F. Haines received his PhD from the University of York in 2009 on shape from shading and stereopsis. Following this he did a post doctorate at Queen Mary, University of London, where he worked on topic models, active learning and background subtraction. He is now at University College London, doing research into handwriting and landscapes. His interests encompass computer vision, machine learning, non-parametric Bayesian methods, belief propagation and density estimation.

Gabriel J. Brostow Gabriel Brostow is an Associate Professor at University College London. He received his Ph.D. and M.S. in Computer Science from the Georgia Institute of Technology in 2004, and his B.S. in Electrical Engineering from the University of Texas at Austin in 1996. He was a Marshall Sherfield Fellow and postdoctoral researcher at the University of Cambridge until 2007, and a research scientist at ETH Zurich until 2009. His research focus is on human-in-the-loop computer vision algorithms for scientists and artists.