BRISKS: Binary Features for Spherical Images on a Geodesic Grid

Hao Guan, and William A. P. Smith
Department of Computer Science, The University of York, UK
{hg607, william.smith}@york.ac.uk

Abstract

In this paper, we develop an interest point detector and binary feature descriptor for spherical images. We take as inspiration a recent framework developed for planar images, BRISK (Binary Robust Invariant Scalable Keypoints), and adapt the method to operate on spherical images. All of our processing is intrinsic to the sphere and avoids the distortion inherent in storing and indexing spherical images in a 2D representation. We discretise images on a spherical geodesic grid formed by recursive subdivision of a triangular mesh. This leads to a multiscale pixel grid comprising mainly hexagonal pixels that lends itself naturally to a spherical image pyramid representation. For interest point detection, we use a variant of the Accelerated Segment Test (AST) corner detector which operates on our geodesic grid. We estimate a continuous scale and location for features and descriptors are built by sampling onto a regular pattern in the tangent space. We evaluate repeatability, precision and recall on both synthetic spherical images with known ground truth correspondences and real images.

1. Introduction

Spherical images arise in a number of contexts. In computer vision, they are captured by omnidirectional cameras such as catadioptric or fisheye imaging systems or by stitching multiple perspective images. Such panoramic images have many applications where their full coverage of the viewing sphere provides a richer source of image features and increases the likelihood of matching features between views. Spherical images have taken on a particular importance recently due to the rapid rise in popularity of "360 video". There are now commercially available systems for capturing such video[1] and support for their interactive playback is available in the most popular online video repositories[2]. Outside computer vision, any discretely sampled spherical function may be processed as a spherical image. The most obvious examples occur in Geographic Information Systems (GIS) which model the spatial variation of properties of the surface of the Earth [21].

Image processing for spherical images is less well developed than for their planar counterpart. There are two complexities that arise in processing spherical images. The first is that the nature of the spherical surface must be taken into account when performing geometric operations. For example, distance between image points depends on geodesic (great circle) distance as opposed to Euclidean distance for planar images. The second is that for discrete images, the spherical surface must be segmented into discrete pixels which can be efficiently indexed. A common means to address both of these issues is to parameterise a spherical image to the 2D plane via a chosen projection (e.g. equirectangular) and then treat the image as if it were planar. The problem with this approach is that any projection introduces distortion and boundaries. A more attractive alternative is to discretise spherical images on the sphere and perform image processing on the surface of the sphere.

One of the most fundamental operations in image processing is to identify points of interest and to build descriptors of local appearance around an interest point. This enables point-to-point matches to be established between images or a compact description of a scene or object to be constructed. The seminal work in this area is the Scale Invariant Feature Transform (SIFT) of Lowe [12]. While SIFT features remain a benchmark for their distinctiveness and robustness to appearance variation, the computational cost of extracting and matching them has led to them being largely superseded by a class of lighter weight features based on binary descriptors. In this paper, we take one such feature, BRISK (Binary Robust Invariant Scalable Keypoints) [11], as inspiration and extend them to operate on spherical images. The proposed BRISKS (BRISK on the Sphere) framework includes a novel discretisation of spherical images and scale space, interest point detection, binary feature description and matching. We perform experimental evaluation on both synthetic and real images and test robustness to rotation, camera motion, illumination changes and noise.

[1] https://vr.gopro.com/
[2] https://support.google.com/youtube/answer/6178631?hl=en-GB

2. Related Work

Feature descriptors for omnidirectional images can be classified into naive methods that use planar descriptors on a planar embedding of the image and those that account for the spherical geometry and build the feature on the sphere.

A number of previous works have taken the former approach and compute planar feature descriptors on a planar embedding of a spherical image (most commonly equirectangular or directly on an image acquired by a catadioptric camera). Goedemé *et al*. [7] and Scaramuzza *et al*. [22] both compute SIFT features directly on an unwrapped spherical image. Benseddik *et al*. [2] apply the planar BRISK descriptor to catadoptric images and find improved performance over SIFT. The drawback of these approaches is that the distortion varies with pose changes and so the local appearance around an interest point may vary dramatically, even with only a change in viewing direction.

An improvement over working directly with a planar embedding is to locally adapt the embedding to reduce the distortion. Fiala *et al*. [6] transform the sphere to a cube map and then compute SIFT features on each cube face. Mauthner *et al*. [14] match regions in omnidirectional images by generating virtual perspective views. Pagani *et al*. [19] regularly distribute points over the sphere and generate a perspective image centred on each point with a fixed field of view. They further introduce affine distortions (in the same manner as Affine-SIFT [18]) providing invariance to both spherical distortion and affine transformations of the scene. Although robust, such an approach dramatically increases computational cost since a large number of images must be generated by resampling and processed independently. Hansen *et al*. [9] and Arican *et al*. [1] both build descriptors based on circular support regions on the sphere centred on an interest point. Hansen *et al*. [9] resample to the tangent plane and then build planar SIFT descriptors. Arican and Frossard [1] build log-polar descriptors on the sphere in a way that accounts for the different sampling densities.

The most attractive approaches avoid issues of distortion and boundaries and work directly on the sphere. We are aware of two previous pieces of work that do so [4, 24]. Cruz-Mota *et al*. [4] provide a full spherical extension of SIFT. This approach handles the differential geometry of the spherical image surface providing highly robust features. As with planar SIFT, the drawbacks are computational cost. In this case, the situation is even worse since computing the scale space via heat diffusion is expensive.

The most relevant previous work to ours is the spherical extension of the ORB descriptor (SPHORB) by Zhao *et al*. [24]. Like us, they work on a hexagonal geodesic grid, use a spherical extension of AST [8] and extend a planar binary descriptor to the sphere (ORB [20] in their case, BRISK [11] in ours). However, there are some important differences. First, they transform the spherical geodesic grid to the plane by partitioning the sphere into sets of triangles. This flattening induces distortion and requires additional processing to handle issues at the boundaries of the triangles. All of our processing is intrinsic to the sphere using geodesic distances and concepts from differential geometry (the log map) in order to build a chart around an interest point. Part of their motivation for flattening is efficient indexing and storage. We represent the grid as a mesh and store it using a halfedge data structure which allows efficient access to local neighbourhoods. The SPHORB grid reduces distance distortion by about one order of magnitude compared to a latitude-longitude image. By storing the image directly in a geodesic grid, our representation has zero distortion. Second, they use a fixed-size AST pattern for interest point detection. We use different sized patterns that differ by a scale factor of 1.5 in order to detect interest points at intra-octaves. Third, their feature descriptor is built directly on the (flattened) hexagonal grid. This means that irregularities caused by pentagonal pixels cannot be handled and features detected in these regions are discarded. We resample local regions onto a standard pattern in the tangent space meaning that we can handle any pixel structure and sub-pixel refined feature locations. Finally, the BRISK framework that we extend includes sub-pixel position refinement and sub-octave scale refinement, whereas SPHORB does not.

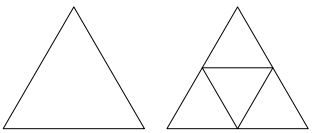
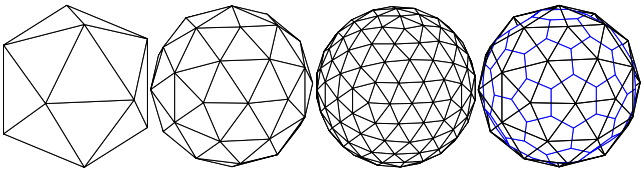


Figure 1: Aperture 4 triangle subdivision rule. By adding additional vertices to the middle of each edge, an equilateral triangle is subdivided into four equally sized triangles.



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

Figure 2: Subdivision process for Quaternary Triangular Mesh. a-c: icosahedron base surface, once subdivided and twice subdivided. d: Dual polyhedron of the QTM (black) is an aperture 4 hexagon grid (blue).

3. Multiscale Geodesic Grid Representation

In this section we describe how images on the S^2 sphere are discretised, stored, processed in a multiscale manner and represented for local feature detection and description.

We segment the sphere into discrete pixels via a process of recursive subdivision. Starting with an icosahedron as a

base shape, we use an aperture 4 triangle hierarchy. This means each triangle is subdivided into four by adding a vertex to the middle of each edge as shown in Figure 1. The newly formed vertices are reprojected to the surface of the sphere. This subdivision provides a triangular segmentation whose surface approximates a sphere with increasing accuracy at each level of subdivision. This is known as a Quaternary Triangle Mesh (QTM) and we show three levels of subdivision in Figure 2.

Our image representation is based on hexagonal pixels which are obtained by taking the dual polyhedron of the triangular mesh, i.e. each vertex in the triangular mesh becomes the centre of a hexagonal face as shown in Figure 2d. It is impossible to completely tile a sphere with hexagons. With an icosahedron as the base shape, the triangular subdivision mesh contains 12 vertices with 5 neighbours (regardless of subdivision resolution). Hence, the hexagonal grid at all resolutions contains 12 pixels that are pentagons. These are handled appropriately throughout our pipeline.

In a planar, rectangular image, the vertical and horizontal resolution (and hence the number of pixels) can be chosen arbitrarily (although often height and width are set equal and made a power of two to ease multiscale processing). On the other hand a spherical image stored as a geodesic grid has a relatively small set of possible resolutions determined by the subdivision scheme and the level of subdivision.

3.1. Storage and indexing

The hexagonal segmentation at subdivision level s and the corresponding spherical image is stored via the corresponding QTM using a triangle mesh $\mathcal{M}_s = (\mathcal{K}_s, \mathbf{V}_s, \mathbf{T}_s)$. The adjacency information is stored in the simplicial complex $\mathcal{K}_s$, whose elements can be vertices $\{i\}$, edges $\{i, j\}$, or faces $\{i, j, k\}$, with indices $i, j, k \in [1\,..\,V_s]$, where V_s is the number of vertices. Each vertex in the mesh corresponds to a hexagonal pixel. The position of each vertex is stored in the matrix $\mathbf{V}_s \in \mathbb{R}^{3\times V_s}$ which contains the 3D coordinates $\mathbf{v}_{s,i} \in \mathbb{R}^3$ of the respective vertices. Since $\mathbf{v}_{s,i}$ are points on the S^2 sphere, $\|\mathbf{v}_{s,i}\| = 1$. The colour of each hexagonal pixel is stored as a matrix of per-vertex colours, $\mathbf{T}_s \in \mathbb{R}^{3\times V_s}$, associated with the triangle mesh. The colour of the ith pixel is written as $\mathbf{t}_{s,i} \in \mathbb{R}^3$ or $t_{s,i} \in \mathbb{R}$ for a grayscale image. Note that when a new image is loaded, only $\mathbf{T}_s$ changes. The mesh structure, adjacency and neighbourhoods are all fixed so can be precomputed and stored.

Throughout the interest point detection and feature description pipeline, we require efficient indexing of pixel neighbours and local neighbourhoods. To ensure that this is possible, we store the QTM in a half-edge data structure [5]. This structure allows vertex adjacency queries to be calculated in $\mathrm{O}(1)$ time. Hence, in asymptotic terms, accessing local neighbourhoods is the same cost for the geodesic grid as for a 2D planar image.

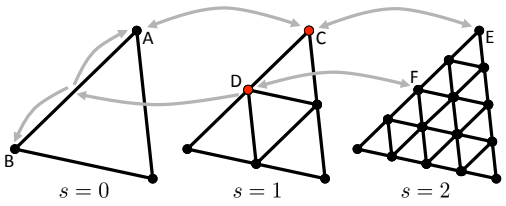


Figure 3: Adjacency across scale: C has neighbours at both finer (E) and coarser (A) subdivision. D has a neighbour at finer subdivision (F) but we consider it to have two neighbours at coarser subdivision (A and B).

Spatial neighbours of a vertex (and hence a hexagonal pixel) i are given by adjacent vertices in the mesh $\mathcal{N}_1(\mathbf{v}_{s,i}) = \{j | \{i, j\} \in \mathcal{K}_s\}$. We write the n-ring neighbourhood of a vertex as $\mathcal{N}_n(\mathbf{v}_{s,i})$. They can either be computed on-the-fly using the half-edge structure or precomputed and stored for fast access.

We also define neighbours across scale. This is used for non-maxima suppression and feature scale refinement. A vertex $\mathbf{v}_{s,i}$ at subdivision level s always has a well-defined neighbour at the next finer subdivision level $s+1$, $\mathcal{N}_1^{s+1}(\mathbf{v}_{s,i}) \in \mathcal{K}_{s+1}$, $|\mathcal{N}_1^{s+1}(\mathbf{v}_{s,i})| = 1$. This is because the vertices of the mesh at subdivision level s are a subset of those at subdivision level $s+1$. On the other hand, at the next coarser level of subdivision, a vertex can have either one or two adjacent neighbours $\mathcal{N}_1^{s-1}(\mathbf{v}_{s,i}) \subset \mathcal{K}_{s-1}$, $|\mathcal{N}_1^{s-1}(\mathbf{v}_{s,i})| \in \{1, 2\}$. This is illustrated in Figure 3.

3.2. Pyramid generation

For scale invariance, we detect features across scale-space and construct descriptors at an appropriate scale. We therefore require an efficient representation of scale-space. We follow the BRISK framework and discretise to a pyramid comprising n octaves and n intra-octaves. Although the scale discretisation is fairly coarse, we estimate a precise feature scale in continuous scale-space by interpolation.

Our subdivision scheme lends itself naturally to construction of a spherical image pyramid. The aperture 4 subdivision corresponds to a halving of scale and hence a one octave shift in scale-space. We construct a scale-space pyramid comprising n octaves c_i for $i \in \{0, 1, \ldots, n-1\}$. The octave count is related to the number of subdivisions of the geodesic grid by $i = s_{\max} - s$, where $s_{\max}$ is the number of subdivisions of the maximally subdivided mesh. In our experiments we use $n = 4$ octaves and $s_{\max} = 8$.

The highest resolution image (corresponding to c_0) is created by sampling a panoramic image onto the maximally subdivided mesh using bilinear interpolation. Successive octaves are formed by area-weighted averaging and subsampling. A down-sampled hexagonal pixel is computed as a weighted average of 7 hexagons, with unit weight for the hexagon contained entirely within the down-sampled

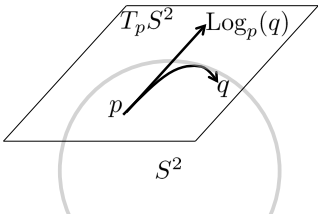


Figure 4: The log map on the S^2 sphere.

hexagon and a weight of 0.5 for the 6 that are half contained. For pentagonal pixels, the same approach is used but there will be only 5 neighbours that are again half contained within the pixel.

To increase resolution in scale, it is desirable to also include intra-octaves. We define n "virtual" intra-octaves d_i that are located in between octaves c_i and c_{i+1}. They are virtual in the sense that we do not compute and store images at the intermediate sizes. Instead, we use an interest point detector that is scaled 1.5 times larger and, when sampling image data for descriptor construction, apply an appropriate scaling to pixel coordinates in the tangent plane. There are two reasons for using virtual intra-octaves. First, there is an efficiency saving in not having to compute and store the intra-octave images. Second, the subdivision scheme cannot produce intra-octaves of appropriate resolution (a scale 1.5 times that of the aperture 4 subdivision can be obtained by an aperture 9 subdivision but the dual polyhedron is not then tile-able with hexagons). The scale t at an octave or intra-octave is given by $t(c_i) = 2^i$ and $t(d_i) = 2^i \cdot 1.5$.

3.3. Tangent space representation

For sub-pixel position refinement and sampling data for descriptor construction, it is useful to transform pixels in the spherical image to a local 2D representation. The S^2 sphere is a Riemannian manifold and hence it is possible to build a chart to parameterise the manifold locally. Since our features describe local appearance only, the locality assumption is reasonable.

We build local charts in the tangent plane to the sphere. We do so using the Riemannian log map, $\text{Log}_p : S^2 \rightarrow T_pS^2$, which transforms from the sphere to the tangent plane at a point $p \in S^2$. The log map linearises the sphere around the base point in such a way that Euclidean distances from the base point in the tangent plane are equal to geodesic distances, i.e. $d_g(p,q) = \|\text{Log}_p(q)\|$. See Figure 4 for a visualisation of the log map.

For the sphere, the log map of a point $q \in S^2$ at base point $p \in S^2$ is given by:

$$\text{Log}_p(q) = \frac{\arccos(\langle p,q\rangle)(q - \langle p,q\rangle p)}{\|q - \langle p,q\rangle p\|}, \qquad (1)$$

Since we deal with unit vectors corresponding to an embedding of the sphere in Euclidean space, the inner product is simply the scalar product of the embedded vectors: $\langle p,q\rangle = \mathbf{p}\cdot\mathbf{q}$, where $\mathbf{p} = \phi(p) \in \mathbb{R}^3$, $\mathbf{q} = \phi(q) \in \mathbb{R}^3$ and ϕ is an arbitrary embedding. The resulting tangent plane vectors are also subject to this arbitrary embedding. So, to yield 2D coordinates in a standardised coordinate system, we apply a rotation to align the plane with the x-y plane and discard the z coordinate as follows:

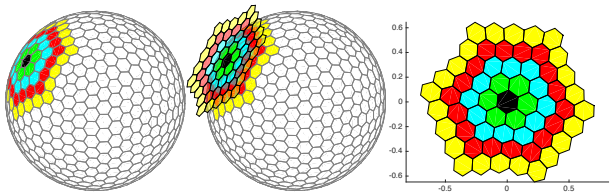
$$\mathbf{x}_p(q) = \mathbf{T}\phi(\text{Log}_p(q)) \in \mathbb{R}^2, \qquad (2)$$

where the transformation is given by

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{R}(\boldsymbol{\alpha}, \theta), \qquad (3)$$

where $\mathbf{R}(\boldsymbol{\alpha},\theta) \in SO(3)$ is a rotation matrix that depends upon $\mathbf{p}$ with rotation axis $\boldsymbol{\alpha} = \mathbf{p} \times [0,0,1]^T = [p_y, -p_x, 0]^T$, and rotation angle $\theta = \arccos(p_z)$. The inverse of the log map is the exponential map, $\text{Exp}_p : T_pS^2 \rightarrow S^2$. This transforms a tangent vector $v \in T_pS^2$ at base point p to the sphere:

$$\text{Exp}_p(v) = p\cos\theta + \frac{v\sin\theta}{\theta}, \quad \theta = \|v\|. \qquad (4)$$



(a) (b) (c)

Figure 5: The log map applied to a hexagonal geodesic grid at subdivision level $s = 3$. (a) An interest point (black) and its 4-ring neighbourhood on the sphere. (b) The local neighbourhood transformed to the tangent plane via the log map. (c) Local 2D coordinate system.

In Figure 5 we visualise the application of the log map to our hexagonal geodesic grid. In (a) we show an interest point (coloured black) and its 4-ring neighbourhood. In (b) we show the neighbourhood transformed to the tangent plane at the interest point via the log map (note that the tangent plane is embedded in 3D space in a way that depends on the base point). Finally, in (c) we show the 2D coordinate system after applying the transformation in Equation 2 to remove the effect of the embedding.

4. Interest point detection

We use the FAST corner detector as the basis of our interest point detection. We begin by describing how the accelerated segment test is extended to operate on a spherical geodesic grid. We then describe how we apply non-maxima

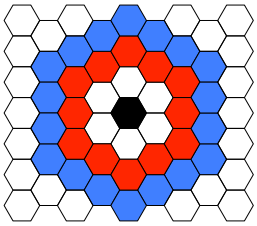


Figure 6: AST pattern for hexagonal lattice. Radius 2 shown in red, radius 3 in blue for corner test at black pixel.

Figure 7: An example of a pixel passing the hexagonal 7-12 AST.

suppression both spatially and across scale in our pyramid representation. Finally, we show how we perform position refinement in space and scale, enabling a continuous estimate of feature position and scale.

4.1. Accelerated segment test on a geodesic grid

In the original FAST approach, a circle of radius 3.4 was discretised onto a square pixel lattice using Bresenham's algorithm. Subsequently, alternate patterns have been considered, approximating circles of radius 1, 2 and 3, which contain 8, 12 and 16 pixels respectively. To test for the existence of a corner, a consecutive sequence of length k pixels must be brighter or darker than the central pixel by a threshold t_{corner}. Originally, k was chosen as 12, corresponding to a 45° corner. This value of k was also computationally efficient since candidate corners could be discounted after testing as few as 3 pixels. Subsequently, it has been found that optimal performance occurs when k is chosen to be the smallest value that will not detect edges. i.e. if the sequence is of length m then $k = \lceil (m+1)/2 \rceil$.

We extend this approach to operate on discrete geodesic grids composed of hexagonal pixel lattices. In general, curves and circles drawn on a hexagonal lattice appear smoother because of the improved angular resolution afforded by having six equidistant neighbours compared to only four in a square lattice [15]. The FAST patterns on a square lattice are only invariant to rotations of 90°, 180° and 270°, whereas the hexagonal patterns are invariant to rotations of 60°, 120°, 180°, 240° and 300°.

We consider circles of radius 2 and 3 on a hexagonal lattice, as shown in Figure 6, containing 12 and 18 pixels respectively. Note that the larger pattern differs in scale (radius) by a factor of 1.5 from the smaller one. We use this fact to detect interest points at intra-octaves. To detect interest points at octave c_i, we use the radius 2 pattern on the geodesic grid at subdivision level $s = s_{\max} - i$. To detect interest points at intra-octave d_i we use the radius 3 pattern, on the geodesic grid at the same subdivision level.

There are a number of special cases to deal with due to the impossibility of completely tiling a sphere with hexagons. Irrespective of level of subdivision, there are 12 pixels with pentagonal shape and hence 5 neighbours. In the radius 2 case, corners centred on pentagonal pixels have circles containing 10 pixels and we test for sequences of length 6. Corners centred on pixels adjacent to a pentagonal pixel have circles of 11 pixels and again we test for sequences of length 6. Finally, the radius 3 case has special cases of circles containing 15, 16 or 17 pixels which we test for sequences of length 8, 9 and 9 respectively. Note that the radius 2 circle shares the same number of pixels as the radius 2 circle on a square lattice. This means that the AGAST [13] decision tree for 12 pixels can be used on a hexagonal lattice without modification.

In Figure 7 we show an example of a pixel passing the hexagonal 7-12 test. The pixel labelled p is regarded as a corner because the 7 pixel sequence (marked in red) lying on the 12 pixel, radius 2 circle are all darker than p by more than the threshold t_{corner}.

Feature saliency can be computed for corners by finding the largest value for the threshold $t_{\text{score}}(p)$ at which the point is still considered a corner. This is done efficiently using binary search on the interval $[t_{\text{corner}}, 1]$.

4.2. Non-maxima suppression

We use the feature saliency score to apply non-maxima suppression. For every detected corner, we compute the corner saliency for adjacent pixels (even if they were not detected as corners) at the same scale. We also compute corner saliency for neighbours across scale (if there are two neighbours at the next coarser scale, we take the one with higher saliency as the neighbour). We require a feature to have maximum saliency amongst all of its neighbours and remove any that do not. The computed saliency scores are saved as they are used subsequently for position refinement.

4.3. Position and scale refinement

We refine the spatial position of each feature to sub-pixel accuracy and estimate a continuous scale. If a feature has been detected at vertex $\mathbf{v}_{s,i}$, we fit a 2D quadratic function in a least squares sense to the set of tangent plane coordinates and FAST scores, $\mathcal{F}(\mathbf{v}_{s,i})$, of the 1-ring neighbourhood of $\mathbf{v}_{s,i}$:

$$\mathcal{F}(\mathbf{v}_{s,i}) = \left([0\ 0]^T, t_{\text{score}}(\mathbf{v}_{s,i})\right) \cup \left\{\left(\mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,j}), t_{\text{score}}(\mathbf{v}_{s,j})\right) | j \in \mathcal{N}_1(\mathbf{v}_{s,i})\right\}. \quad (5)$$

We compute the position and FAST score of the maxima of the quadratic function in closed form and transform the resulting point back to the sphere using the inverse of the transformation applied in Equation 2 followed by the exponential map. We perform the same sub-pixel position refinement on neighbourhoods at adjacent scales, i.e. $\mathcal{F}(\mathbf{v}_{s+1,i})$ and $\mathcal{F}(\mathbf{v}_{s-1,i})$. For these neighbourhoods, it is possible that the central pixel is not the maximum or that the fitted quadratic is not even concave (and hence has no maximum). In these cases, we simply take the position and score of the pixel in the neighbourhood with maximum score.

We now have sub-pixel refined positions and scores for the detected feature and for scales either side of the detected feature. Finally, we estimate a continuous scale by fitting a parabola to the scores as a function of scale, compute the scale of the maximum of the parabola and interpolate the refined position between the position at the original scale and that in the scale direction of the parabola maximum. The refined scale and position are assigned to the feature.

5. Descriptor Extraction

The final step in the pipeline is to generate feature descriptors for the detected features. Following the BRISK framework, this is done in four steps. First, we make an estimate of the local gradient direction in order to assign a direction to the feature. Second, the local neighbourhood is rotated to normalise this direction (introducing rotation invariance). Third, the local neighbourhood is sampled onto a fixed sampling pattern. Finally, the binary descriptor is generated based on a fixed set of intensity comparisons on this descriptor.

5.1. Orientation normalisation

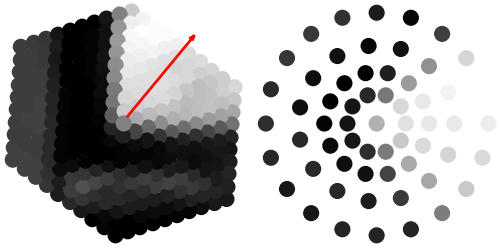
We begin by computing an estimate of the local gradient direction for each feature. To do so, we use the 9-ring neighbourhood around the feature point at the octave in which the feature was detected. Suppose that a feature was detected at vertex $\mathbf{v}_{s,i}$. The gradient according to a pair of pixels $\mathbf{v}_{s,j}$ and $\mathbf{v}_{s,k}$, $j,k \in \mathcal{N}_{1..9}(\mathbf{v}_{s,i})$, in the neighbourhood of the feature is computed in the tangent plane by:

$$\mathbf{g}(\mathbf{v}_{s,j},\mathbf{v}_{s,k}) = \tag{6}$$
$$\mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,k}) - \mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,j}) \cdot \frac{t_{s,k} - t_{s,j}}{\|\mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,k}) - \mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,j})\|^2}.$$

The estimate of the overall characteristic direction:

$$\mathbf{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{|\mathcal{L}|} \sum_{(j,k)\in\mathcal{L}} \mathbf{g}(\mathbf{v}_{s,j},\mathbf{v}_{s,k}) \tag{7}$$

is computed by averaging the local gradient direction estimates between all "long distance" pairs. That is the set of pairs, $\mathcal{L} = \{(j,k) \mid j,k \in \mathcal{N}_{1..9}(\mathbf{v}_{s,i}) \wedge \|\mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,k}) - \mathbf{x}_{\mathbf{v}_{s,i}}(\mathbf{v}_{s,j})\| > \delta_{\min}\}$, whose Euclidean distance is greater than a threshold, $\delta_{\min}$. We choose the threshold to be the radius of the neighbourhood, defined as the point on the 9-ring that is minimum distance from the base point. The motivation for using long distance pairs in the original BRISK framework is that local gradients annihilate each other and therefore do not influence the global gradient estimate. In Figure 8a we show an example of the intensities in the neighbourhood around a feature and the estimated characteristic direction. Orientation is normalised by applying a rotation of angle $\arctan2(g_y, g_x)$ to the tangent plane coordinates of the pixels in the local neighbourhood.



(a) Characteristic orientation. (b) Sampled intensities.

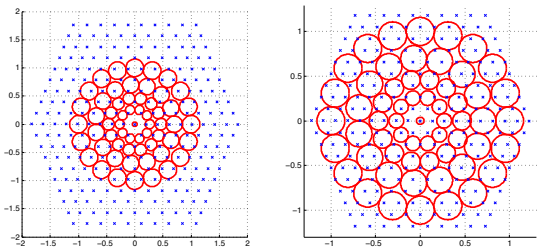
Figure 8: Orientation normalisation and sampling. Left: a neighbourhood around a feature. Right: the intensities after sampling the rotated neighbourhood onto the standard pattern.

5.2. Sampling

In order to construct feature descriptors, we sample the rotation-normalised intensities onto a standard pattern. This serves numbers of purposes. First, we can deal with any irregularities in pixel structures caused by pentagonal pixels. Second, the sampling uses Gaussian smoothing which reduces aliasing effects. Third, it provides a standardised set of image locations from which a fixed set of intensity comparisons can be used to create feature descriptors.

We use the same pattern as in the BRISK framework. However, note that we are operating in the tangent space to the sphere rather than on a 2D image. The sampling pattern is shown in Figure 9 and comprises a circle of radius 1 with 60 sample points, $\mathbf{s}_1,\ldots,\mathbf{s}_{60} \in \mathbb{R}^2$. Sample points are plotted as red circles and the radius of the circle corresponds to the standard deviation, $\sigma_1,\ldots,\sigma_{60}$, of the Gaussian used for smoothing at that sampling point. We denote the smoothed, sampled intensity at sample point $\mathbf{s}_i$ as $I(\mathbf{s}_i,\sigma_i)$. After rotation normalisation and sampling, the image region (Figure 8a) results in the sampled intensities (Figure 8b). It is from these intensities that we compute the intensity comparisons to build the feature descriptor.

Pixel neighbourhoods are scaled according to the scale of the detected feature prior to sampling. Since we do not explicitly compute intra-octave images, intra-octave features



(a) Sampling octave feature. (b) Sampling intra-octave feature.

Figure 9: The sampling pattern consists of points (shown as red circles) distributed over a radius 1 circle. The 9-ring neighbourhood around a feature is scaled onto the pattern as shown (shown as blue crosses).

Figure 10: Subset of the SUN360 dataset [23] used in our experiments.

are scaled by a factor of 1.5 on the octave image at which they were detected. This is illustrated in Figure 9.

5.3. Descriptor generation

The bit string descriptor is built using intensity comparisons on a set of short-distance pairs $\mathcal{S}\{(i,j) \mid i,j \in \{1,\dots,60\} \wedge \|\mathbf{s}_i - \mathbf{s}_j\| < \delta_{\max}\}$. The motivation for only using comparisons between pairs of locations that are spatially close is that feature similarity then only requires brightness variations to be locally consistent. This reduces sensitivity to spatially varying illumination. The short distance threshold, $\delta_{\max}$, is chosen to yield a bit string of the desired length. We follow BRISK [11] and BRIEF64 [3] and use 512 bit descriptors. For our pattern, this corresponds to a value of $\delta_{\max} = 0.6378$. Pairs are evaluated in an arbitrary but fixed order, $S_1 \in \mathcal{S}, \dots, S_{512} \in \mathcal{S}$, yielding the bit string descriptor b, where

$$b_i = \begin{cases} 1 & \text{if } I(\mathbf{s}_j, \sigma_j) < I(\mathbf{s}_k, \sigma_k) \\ 0 & \text{otherwise} \end{cases} \quad \text{where } S_i = (j,k). \tag{8}$$

The dissimilarity between two feature descriptors is given by their Hamming distance. Since this amounts to nothing other than a bitwise XOR followed by a bit count, this can implemented very efficiently.

6. Experimental results

We evaluate our proposed BRISKS features on both synthetic rendered images and real images. The use of synthetic images allows us to arbitrarily control illumination and also means that ground truth correspondences can be computed for images taken from different viewpoints. We include comparison to two previous methods. The first applies classical planar SIFT to unwrapped equirectangular panoramic images, as done by [7, 22]. The second is the spherical extension of SIFT introduced by Cruz-Mota *et al.* [4]. We use standard feature evaluation metrics [16, 17] adapted to the sphere. Feature detection performance is measured using repeatability [17]. Feature description and matching performance is measured using $1-$precision and recall curves, giving a metric that is invariant to the matching threshold. The synthetic images are rendered using the spherical camera sensor in the Mitsuba renderer [10]. We use the Babylonian City scene from the Medieval City collection (courtesy of Johnathan Good). We render both panoramic images and depth maps. The real images we use are a 10 image subset of the SUN360 dataset [23], as shown in Figure 10, spanning a range of different scene types. We resize input images for SIFT and SSIFT so that the number of pixels is approximately equal to the number of pixels in our finest subdivision mesh.

6.1. Detector repeatability

The repeatability score measures the ability of the interest point detector to detect features at image points in different images corresponding to the same scene location. For each detected interest point in the first image, we project the point into the scene, project it back into the second image and check whether an interest point has been detected at the corresponding location. Our criteria for a successful repeated detection is simply to measure the spherical distance between the reprojected and closest detected interest points and test whether it smaller than a threshold of $2°$. The repeatability score is the ratio between the number of repeat detections and the smaller of the number of interest points in the pair of images.

In synthetic images, we use the ground truth depth map to project image points into the scene allowing us to compute correspondence even with viewpoint changes. For the real images we only apply rotations and add noise and hence simply need to rotate image points to test for repeated detections. For all three methods, we adjust the detection threshold to yield similar numbers of interest points in each image.

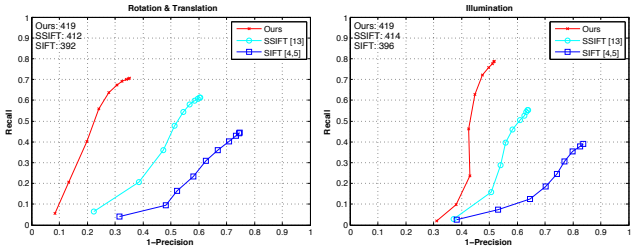
In Table 1 we show repeatability results for the synthetic images. The second column shows results for pose changes (i.e. the rotation and translation of the camera differs between views). We rendered 6 images in which the camera follows a linear trajectory with a distance of 50 units along the x axis between each image. We also apply a rotation about the z direction of $60°$ between each image. We measure repeatability between pairs of consecutive images and show results averaged over all pairs. In the third col-

	Rotation and Translation	Illumination
Ours	**0.93** (419)	**0.64** (414)
SSIFT [4]	0.65 (412)	0.49 (414)
SIFT [7, 22]	0.57 (392)	0.41 (396)

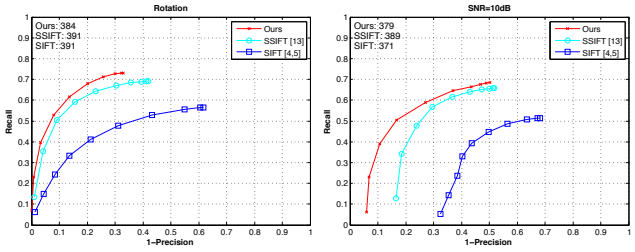
Table 1: Repeatability on synthetic images. Average number of detected features shown in brackets.

	Rotation	Noise		
		10dB	15dB	20dB
Ours	**0.94** (384)	**0.90** (379)	**0.93** (383)	**0.93** (385)
SSIFT [4]	0.86 (391)	0.76 (389)	0.79 (379)	0.83 (374)
SIFT [7, 22]	0.70 (391)	0.65 (371)	0.67 (378)	0.66 (384)

Table 2: Repeatability on SUN360 images. The noise level is shown as signal to noise ratio. Average number of detected features shown in brackets.



(a) Under rotation & translation and illumination



(b) Under pure rotation and noise

Figure 11: 1−precision and recall curves a: for the synthetic images and b: for real data. The numbers of correspondences are shown on the left corner in each figure.

umn we show results for changing illumination. For three viewpoints, we render images with illumination simulating 12 noon and 5:00 pm in the afternoon with a 60° rotation about the z axis between images. We measure repeatability between each pair and average results over all pairs.

In Table 2 we show repeatability results for real images. The second column shows results for rotation. For each scene we generate six images differing by rotations of 60° about the z axis. Columns three to five show results with additive noise. We add Gaussian noise to each image with the variance selected to obtain a desired signal to noise ratio.

6.2. Descriptor precision and recall

We evaluate our descriptor on the same sets of images as for the detector evaluation. We do so using the standard 1−precision and recall metrics. For each feature in the first image we find the nearest neighbour feature descriptor in the second image (using Hamming distance for our descriptors and Euclidean distance for SIFT and SSIFT). We vary matching threshold and plot how precision and recall vary. Recall is the ratio between the number of correct matches and the number of feature pairs that have correspondences. 1−precision is the ratio between the number of incorrect matches and the total number of matches. Correct matches are defined in the same way as in the repeatability score.

Figure 11 shows the averaged curves for the synthetic images and real images. Rotation and translation results are on the (a) left, illumination results on the (a) right. Note that these are very challenging images since there are repeated texture patterns (e.g. bricks and paving) and the changes in pose and illumination cause dramatic changes in appearance. Results for rotation only are shown on the (b) left, results for additive noise are shown on the (b) right. These images are less challenging than the synthetic ones since camera position and illumination is fixed.

7. Conclusions

In this paper we have proposed a binary feature for spherical images. This requires rethinking a number of fundamental aspects of a local feature such as how an image is discretised and stored, how to build a discrete scale space and how to perform feature detection and description without having to project the spherical image to a planar embedding. Despite having significantly lower computational complexity than SIFT-like methods and a descriptor that is 16 times smaller (for 128D SIFT descriptor with double precision floats, or 8 times smaller for single precision), across all experimental conditions on both synthetic and real images, our method significantly outperforms SSIFT which, in turn, outperforms the naive application of SIFT to equirectangular images.

There are a number of ways that this work could be extended. First, there is scope to explore alternative sampling patterns and comparison pairs, exploiting any developments in 2D feature description that may improve performance. Second, we would like to apply our features to applications such as structure-from-motion with panoramic images or realtime visual odometry with omnidirectional cameras. Third, the approach could be extended to dense matching to enable applications such as motion segmentation, optical flow or stereo to be addressed. Finally, an interesting alternate approach would be to extend deep learning methods to our spherical image representation. For example, a CNN could operate on our geodesic grid with convolution operations taking place on the sphere. This would allow us to learn features on the sphere that may be appropriate for higher level visual tasks.

References

[1] Z. Arican and P. Frossard. Sampling-aware polar descriptors on the sphere. In *Proc. ICIP*, pages 3509–3512, 2010. 2

[2] H. E. Benseddik, H. Hadj-Abdelkader, B. Cherki, and O. Djekoune. Binary feature descriptor for omnidirectional images processing. In *Proc. IPAC*, 2015. 2

[3] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proc. ECCV*, pages 778–792, 2010. 7

[4] J. Cruz-Mota, I. Bogdanova, B. Paquier, M. Bierlaire, and J.-P. Thiran. Scale invariant feature transform on the sphere: Theory and applications. *Int. J. Comput. Vis.*, 98(2):217–241, 2012. 2, 7, 8

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer Verlag, Berlin, 1997. 3

[6] M. Fiala and G. Roth. Automatic alignment and graph map building of panoramas. In *Proc. IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, 2005. 2

[7] T. Goedemé, T. Tuytelaars, L. Van Gool, G. Vanacker, and M. Nuttin. Omnidirectional sparse visual path following with occlusion-robust feature tracking. In *Proc. Workshop on omnidirectional vision, camera networks and non-classical cameras (OMNIVIS)*, 2005. 2, 7, 8

[8] H. Guan and W. A. P. Smith. Corner detection in spherical images via accelerated segment test on a geodesic grid. In *Proc. ISVC*, pages 407–415, 2013. 2

[9] P. Hansen, P. Corket, W. Boles, and K. Daniilidis. Scale invariant feature matching with wide angle images. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1689–1694, 2007. 2

[10] W. Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 7

[11] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proc. ICCV*, pages 2548–2555, 2011. 1, 2, 7

[12] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, 2004. 1

[13] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proc. ECCV*, pages 183–196, 2010. 5

[14] T. Mauthner, F. Fraundorfer, and H. Bischof. Region matching for omnidirectional images using virtual camera planes. In *Proc. Computer Vision Winter Workshop*, 2006. 2

[15] L. Middleton and J. Sivaswamy. *Hexagonal Image Processing*. Springer, 2005. 5

[16] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005. 7

[17] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comput. Vis.*, 65(1-2):43–72, 2005. 7

[18] J.-M. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM J. Imaging Sci.*, 2(2):438–469, 2009. 2

[19] A. Pagani, C. Gava, Y. Cui, B. Krolla, J.-M. Hengen, and D. Stricker. Dense 3d point cloud generation from multiple high-resolution spherical images. In *Proc. International Conference on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, pages 17–24, 2011. 2

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. ICCV*, pages 2564–2571, 2011. 2

[21] K. Sahr, D. White, and A. J. Kimerling. Geodesic discrete global grid systems. *Cartogr. Geogr. Inf. Sci.*, 30(2):121–134, 2003. 1

[22] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Trans. Robotic. Autom.*, 24(5):1015–1026, 2008. 2, 7, 8

[23] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *Proc. CVPR*, pages 2695–2702, 2012. 7

[24] Q. Zhao, W. Feng, L. Wan, and J. Zhang. SPHORB: a fast and robust binary feature on the sphere. *Int. J. Comput. Vis.*, 113(2):143–159, 2015. 2