Discovering Details and Scene Structure with Hierarchical Iconoid Shift

Tobias Weyand and Bastian Leibe
Computer Vision Group
RWTH Aachen University, Germany
{weyand,leibe}@vision.rwth-aachen.de

Abstract

Current landmark recognition engines are typically aimed at recognizing building-scale landmarks, but miss interesting details like portals, statues or windows. This is because they use a flat clustering that summarizes all photos of a building facade in one cluster. We propose Hierarchical Iconoid Shift*, a novel landmark clustering algorithm capable of discovering such details. Instead of just a collection of clusters, the output of HIS is a set of dendrograms describing the detail hierarchy of a landmark. HIS is based on the novel* Hierarchical Medoid Shift *clustering algorithm that performs a continuous mode search over the complete scale space. HMS is completely parameter-free, has the same complexity as Medoid Shift and is easy to parallelize. We evaluate HIS on 800k images of 34 landmarks and show that it can extract an often surprising amount of detail and structure that can be applied,* e.g.*, to provide a mobile user with more detailed information on a landmark or even to extend the landmark's Wikipedia article.*

1. Introduction

Current landmark recognition approaches, such as [1, 5, 9, 12, 16, 25, 27] or Google Goggles typically only return the name of the photographed building. While this can be useful for internet image annotation [16, 9], it is actually of very little use for a mobile user. Imagine a tourist visiting Paris' major landmarks to see Notre Dame Cathedral (see Fig. 1). He already knows that he is at Notre Dame – that was the purpose of his trip, after all. In addition, his smartphone has a GPS sensor that can tell him similar information. We believe that an image-based recognition engine can only bring a benefit in such mobile scenarios if it can deliver much more fine-grained information, such as annotations of building details, statues, ornaments, or paintings. However, current landmark discovery approaches are not up to this task yet, and it is an open question how such a detailed structure can be automatically discovered in image collections.

Previous work has considered the task as a clustering problem, producing either hard [9, 15, 16] or soft [25] clusters. While some methods are able to discover objects at finer scales, they are dependent on parameters that define the scale of the objects to be discovered in advance [25].

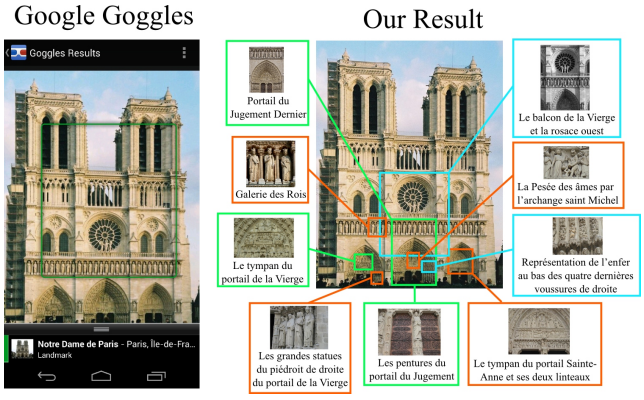


Figure 1: In contrast to previous landmark clustering approaches, Hierarchical Iconoid Shift also discovers scene details that are part of larger structures. These details make up nodes in the tree-structured scene description returned by the algorithm.

We propose a novel algorithm for clustering internet photo collections, called *Hierarchical Iconoid Shift*, that is able to discover objects of interest at any scale. It is based on *Hierarchical Medoid Shift*, a novel variant of Medoid Shift [19] inspired by scale space theory that tracks local density maxima while continuously increasing the kernel bandwidth. Hierarchical variants of Mean Shift have been proposed before [6, 24], but they proceed in discrete bandwidth steps. In contrast, our algorithm increases the bandwidth continuously and is thus completely parameter-free. HMS discovers modes at all scales and constructs a dendrogram from their merging behavior. It has the same runtime and memory complexity as Medoid Shift and is just as easy to parallelize.

We apply HMS to the task of clustering internet photo collections using the Iconoid Shift framework [25] and show that it discovers the structure of a scene starting at small-scale objects such as individual statues or ornaments up to the full view of the landmark building. HIS produces a set of iconoic images, or *Iconoids*, as well as their corresponding clusters at any kernel bandwidth. Iconoids are organized in a dendrogram structure in which a path from

leaf to root represents the hierarchy of iconic views for a particular photo.

We evaluate our approach on 34 historic landmarks with a rich detail structure. Additionally, we show that by matching the resulting Iconoids against the details depicted in a landmark's Wikipedia page, Iconoids can be accurately annotated with semantic labels. Simultaneously, this matching provides information on what interesting objects are still missing from the Wikipedia description and where they are located with respect to already annotated structures.

Related Work. Many different clustering methods have been applied to discover landmarks in internet photo collections. Early approaches used algorithms that produce a non-overlapping clustering like Spectral Clustering [15] or Single-Link Hierarchical Agglomerative Clustering [9, 16, 27]. Later work has shifted to producing a soft or overlapping clustering of landmarks by using LDA [14], PLSA [20], Kernel Vector Quantization [1] or Medoid Shift [25].

Multi-scale descriptions of internet photo collections have so far received less attention. [5] apply Mean Shift to the geotags of photos at different bandwidths representing city scale and landmark scale. [7] create an accumulated top-down view of the view frustums of all cameras in a scene called the "geo-relevance image" and cluster it hierarchically by applying Mean Shift in a top-down fashion. As for our approach, the result of their method is a tree of clusters describing the scene. However, since their approach operates on a 2D top-down view of the scene and uses discrete scale steps for clustering, the detail granularity it can discover is limited.

There have been different approaches employing metadata from the web in order to name the discovered landmarks and provide additional information on them. [23] propose a 3D landmark viewer that enables manual annotation of building parts. [17] automate this step by performing web searches for all noun phrases in the Wikipedia article of a landmark and matching the retrieved images against its SfM reconstruction. [20] segment the 3D point cloud of a scene using PLSA and assign frequent Flickr tags to parts of the point cloud likely making up a semantic object. The large majority of approaches uses user-provided tags from internet photo collections [5, 9, 16, 21, 27]. [16] use frequent tags as search queries to find the corresponding Wikipedia article for each landmark. [27] mine online tourist guides to find additional landmarks.

Several approaches to create a hierarchical clustering based on Mean Shift have been proposed. A common idea [6, 24] is to iteratively run Mean Shift, feeding the output of one iteration into the next one as seeds and increasing the bandwidth for each run. Our approach is similar to [11] who also adapt the idea of Scale-Space filtering [26] to clustering by increasing kernel scale and tracking maxima of the kernel density of the dataset using a Mean-Shift-like procedure. The maxima trace out branches in a 3D (x, y, scale) dendrogram. In contrast to their algorithm, HMS is applicable to arbitrary metric spaces and does not use discrete scale increments, but continuously sweeps over all scales, making it completely parameter-free. Iconoid Shift (IS) [25] applies Medoid Shift to image collections, but uses a fixed bandwidth. We use the IS framework to apply HMS to the task of finding details and scene structure in photo collections.

2. Hierarchical Medoid Shift

We now first introduce the Hierarchical Medoid Shift (HMS) algorithm and then show how we can adapt the Iconoid Shift framework [25] to apply it to internet photo collections. We focus on Medoid Shift, but the idea of HMS is straightforward to adapt to Mean Shift [3] as well.

Medoid Shift. Mean Shift [3] is a mode search algorithm that, after initialization with a *seed* point, iteratively shifts a kernel window to the mean of the points inside it until convergence at a mode. By weighting points with a kernel φ_β with bandwidth β, Mean Shift implicitly searches for maxima of a kernel density. The kernel

$$\Phi_\beta(d) = \int_0^d \varphi_\beta(e)\mathrm{d}e \tag{1}$$

defining this kernel density is called the *shadow* of φ_β. A set of points is clustered by seeding the algorithm once with each point and associating each point with its mode. The only change Medoid Shift [19] makes to this procedure is that the kernel window is iteratively shifted to the *medoid* of the points inside it, which makes it applicable to arbitrary metric spaces. Given a set of points $\{\mathbf{x}_i\}$ and a metric distance function d, and starting at the current medoid $\mathbf{y}_k$, Medoid Shift finds the next medoid $\mathbf{y}_{k+1}$ by minimizing

$$\mathbf{y}_{k+1} = \operatorname*{arg\,min}_{\mathbf{y}\in\{\mathbf{x}_i\}} \left\{ \sum_i \mathrm{d}(\mathbf{y}, \mathbf{x}_i)\varphi_\beta(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_i)) \right\}. \tag{2}$$

Hierarchical Medoid Shift. A problem with Mean Shift and Medoid Shift is that the choice of kernel bandwidth depends on the data and application. Often, the data has modes of different scales, even making a single fixed bandwidth unsuitable. One way to address these issues is to use a variable bandwidth kernel [4], which however requires the choice of a suitable density estimator and a function for choosing the bandwidth based on the estimated density. Another idea is to run Mean Shift coarse-to-fine at discrete bandwidth steps $\beta_1, \ldots, \beta_N$, using the modes of the run at β_i as seeds for the run at β_{i+1} [6, 24]. However, this approach requires the choice of suitable bandwidth steps. Too large steps might lead to missed modes while too small steps will cause unnecessarily high computational effort.

In this work, we take inspiration from Scale Space Filtering [11, 26] that creates a hierarchical description of a

Algorithm 1 Hierarchical Medoid Shift.

// Input: Point set $\{\mathbf{x}\}$, seed point $\mathbf{y}_0$
// Output: Sequence of modes and bandwidths $S = ((\beta_k, \mathbf{y}_k))$
$k := 0$, $\beta_0 := 0$, $S = ((\beta_0, \mathbf{y}_0))$
while $\exists \mathbf{x} : \mathrm{d}(\mathbf{x}, \mathbf{y}_k) > \beta_k$ **do**
  $\beta_{k+1} := \min_{\mathbf{x}:\mathrm{d}(\mathbf{x},\mathbf{y}_k)>\beta_k} \{\mathrm{d}(\mathbf{x}, \mathbf{y}_k)\}$ // Grow kernel
  $k := k + 1$
  // Perform Medoid Shift
  $\hat{\mathbf{y}} := \emptyset$
  while $\hat{\mathbf{y}} \neq \mathbf{y}_k$ **do**
    $\hat{\mathbf{y}} := \mathbf{y}_k$
    $\mathbf{y}_k = \arg\min_{\mathbf{y}\in\{\mathbf{x}\}} \left\{\sum_{\mathbf{x}} \mathrm{d}(\mathbf{y}, \mathbf{x})\varphi_{\beta_k}(\mathrm{d}(\mathbf{y}_k, \mathbf{x}))\right\}$
  Append $(\beta_k, \mathbf{y}_k)$ to S

signal by filtering it with a continuously growing Gaussian kernel and constructing a tree of its merging extrema. [11] define a clustering approach based on Scale Space Filtering that uses a method similar to Mean Shift to follow modes, but their approach is only applicable to 2D images. Our proposed algorithm allows the application of Scale Space Filtering to arbitrary metric spaces by using Medoid Shift [19] to explicitly track density maxima while continuously increasing the kernel bandwidth. This continuous growing is made possible through the following key observations that to our knowledge have not been made in the literature. (1) If the kernel has finite support, as is common in Medoid Shift, there is only a finite number of bandwidth steps at which a new data point enters a medoid's kernel window. (2) If the kernel also has a monotonously decreasing profile, the weighted distances of the data points to their respective modes change between these steps, but their order remains the same, meaning that the density maxima can only change when a new data point enters the kernel window. This allows for continuously growing the kernel by examining only a finite number of discrete steps.

The Hierarchical Medoid Shift algorithm proceeds as follows: We start from a seed point at kernel bandwidth 0 and build a priority queue of its nearest neighbors, ordered by their distances to the seed. These distances define the discrete steps in which the kernel window grows. In each step, we pop an element from the priority queue, increase the kernel bandwidth to its distance from the medoid and compute its distances to all points inside the kernel window. We then find the new medoid by minimizing Eq. (2). If the medoid has shifted, we update the priority queue with the new neighbor distances. We repeat this procedure until the queue is empty. The detailed algorithm is given in Alg. 1.

For each bandwidth increment, we compute the distances of the new data point with all other points. Thus, in total, the number of distance computations performed is quadratic in the number of points under the final kernel window. The computational complexity is therefore the same as for standard Medoid Shift. However, in contrast to Medoid Shift, HMS is completely parameter-free.

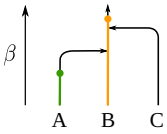


Figure 2: Parallel HMS. Two worker threads (green and orange) are initialized with seeds A and B. When the green worker collides with B, it is finished and can start processing seed C.

The algorithm is initialized once with every data point. Similar to the evolution of local maxima in scale space [26], medoids corresponding to small maxima will merge to form larger maxima. The resulting convergence sequences therefore form a dendrogram of the density structure of the dataset at all scales. A horizontal slice through this dendrogram yields the set of medoids at a particular scale.

Parallelization. HMS can be parallelized by initializing each thread with one seed. We can imagine each run of the algorithm as tracing out a branch of the dendrogram from bottom to top, where we move upwards each time we increase the kernel bandwidth β and we move horizontally each time we shift (Fig. 2). We keep track of the dendrogram branches in a central data structure, using mutexes to prevent race conditions when accessing them. As soon as one branch collides with another, we know that it will follow the same path. Therefore, we stop its corresponding worker thread and re-initialize it with a new seed.

3. Hierarchical Iconoid Shift

We apply HMS to the task of clustering collections of landmark images using the Iconoid Shift framework [25]. We briefly review the key ideas of Iconoid Shift and then show how HMS can be used in this framework.

Iconoid Shift defines a metric space over images based on their overlap and uses Medoid Shift to find modes in this space. These modes, called *Iconoids*, are the photos that have the highest overlap with other photos of the same building or object. Iconoids are typically frontal, centered views in which the landmark fills most of the image [25].

In IS, the *support set* of an image is the set of images under its kernel window, meaning the images that have a certain minimum overlap with it. Formally, an Iconoid is an image that has minimal *homography overlap distance* to the images in its support set. This distance is 0 if two images show exactly identical views and 1 if they have no overlap. In the underlying Medoid Shift algorithm, a hinge function is used as the shadow kernel Φ_β, and thus the kernel φ_β is a step function that cuts off at the bandwidth β, *i.e.* the overlap distance threshold of a support set:

$$\varphi_\beta(d) \quad = \quad \frac{1}{\beta} \text{ if } d < \beta \text{ , 0 otherwise.} \qquad (3)$$

Iconoid Shift computes overlaps using a local matching graph over the images in a support set. While the algorithm progresses, this graph is built on-the-fly by recursively performing retrieval using a visual word based image retrieval engine [13, 22]. Image pairs with at least 15 inliers w.r.t. a homography are connected. The overlap of two images

(i, j) is determined by computing the areas of the bounding boxes around the inlier features in both images. Their overlap is then defined as the smaller of these areas. The *homography overlap distance* is simply one minus the overlap.

$$\mathrm{d}_{ovl}(i,j) = 1 - \min\left\{\frac{||x_{ji}||}{||R_i||}, \frac{||x_{ij}||}{||R_j||}\right\} \quad (4)$$

Here $||x_{ij}||$ and $||x_{ji}||$ are the overlap region sizes and $||R_i||$ and $||R_j||$ the image areas of image i and j respectively.

However, because images with low overlap are usually not directly connected in the graph, missing overlaps are determined using the *homography overlap propagation* (HOP) algorithm [25]. We first build the minimum spanning tree (MST) of the matching graph of a support set. Then, HOP computes the overlap between each pair of images (i, k) by propagating the overlap region of i along the unique path to k in the MST. For example, if (i, j) and (j, k) are edges in the MST, $\mathrm{d}_{ovl}(i, k)$ is computed by projecting i's overlap region with j into j using the homography between i and j, intersecting the resulting region with j's overlap region with k and finally projecting the region into k. Using the known overlap regions, the overlap is then computed using Eq. (4).

Given a seed image, Iconoid Shift explores its support set and builds the local matching graph, computes the pairwise overlaps in the support set as described above, and chooses the images with the highest weighted overlap according to Eq. (2). This process is then repeated until convergence.

Note that Iconoid Shift defines clusters differently from Medoid Shift: In Medoid Shift, a cluster is the set of images converging to the same mode. If this definition were used in Iconoid Shift, a cluster might contain images having no visual similarity to the Iconoid. Instead, Iconoid Shift defines a cluster as the support set of an Iconoid, *i.e.* the set of all images having at least an overlap of $1 - \beta$ with it.

Hierarchical Iconoid Shift. Since naively plugging in the homography overlap distance into Alg. 1 would be very expensive, we now derive an efficient algorithm that applies HMS in the Iconoid Shift framework and that takes advantage of the MST structure and HOP.

We define the *corona* as the images whose overlap distance to the medoid is greater than β, but that match at least one image from the support set (*i.e.* the blue images in Fig. 3a). The overlaps of the corona images with the medoid define the discrete scale steps for growing the kernel. The corona images are inserted into a priority queue and prioritized by their overlap with the medoid. HIS maintains a minimum spanning tree of all images in the corona and support set. New images are added to this tree incrementally using a linear time algorithm [2].

HIS is initialized with a single seed image in the support set and $\beta = 0$. It then queries the retrieval engine with the seed and adds its matching images to the corona. The overlaps of the new corona images with the seed are computed and the images are added to the priority queue.

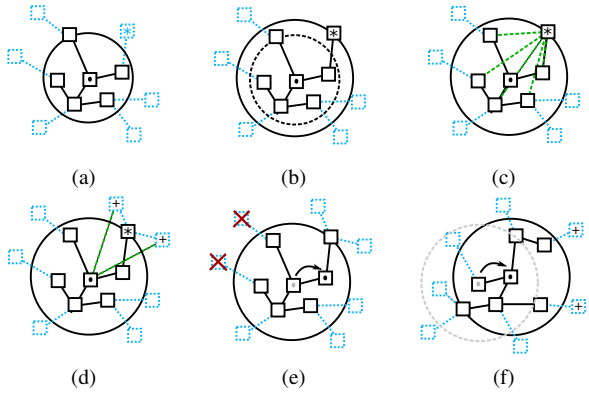


Figure 3: Hierarchical Iconoid Shift. (a,b) The corona image closest to the medoid is added to the support set. The kernel bandwidth is expanded to its distance to the medoid. (c) The pairwise distances between the new image and all other support set images are computed. (d) Images matching the new image are retrieved, inserted into the corona and added to the priority queue. (e,f) If the medoid has shifted, the support set and corona are updated.

An iteration of HIS is shown in Fig. 3. First, we take the top image from the priority queue (Fig. 3a) and increase the kernel bandwidth to the image's overlap distance from the medoid (Fig. 3b). Then, we compute the image's pairwise distances to all images in the support set by propagating its overlap through the MST using HOP (Fig. 3c). Next, we complete the corona by retrieving images matching the newly added image, computing overlaps with the medoid using HOP, and inserting them into the corona and priority queue (Fig. 3d). Finally, we check whether the medoid has shifted (Eq. (2)). If not, we directly continue with the next growing step. If yes, we alternately update the support set and corona (Fig. 3e, 3f) and perform shifting.

The support set update is done in three steps: 1. We move support set images whose overlap distance from the new medoid is larger than β to the corona. 2. We remove corona images with no match in the support set. 3. While there are images in the corona whose overlap distance to the medoid is smaller than β, we add them to the support set, compute their overlaps with all support set images and retrieve new corona images as above. The algorithm is finished when the corona is empty, *i.e.* all images overlapping with the medoid are in the support set. HIS can easily be parallelized in the same way as HMS (Fig. 2).

Since adding an image to a support set has linear complexity in the number of images in it, the total complexity of growing a branch is quadratic in the number of images in its final support set. Since we can stop processing a branch if a worker merges into an existing branch, the overall runtime of HIS depends on the branching factor of the dendrogram,

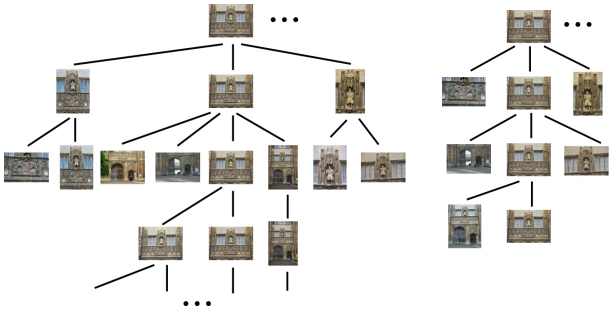


Figure 4: Part of a dendrogram from Trinity College before (left) and after (right) simplification.

and thus on the density distribution in the dataset. We observed that typically, there are a few prominent Iconoids that grow for a large range of bandwidths and that most other branches merge into after a few growing steps.

Dendrogram Simplification. The dendrograms produced by Hierarchical Iconoid Shift often have some redundancy, *e.g*. because of drastic illumination changes preventing images of the same object to match, which causes the formation of multiple Iconoids (see Fig. 4 (left) for an example).

Since these redundant dendrogram branches usually show very little change in perspective, we use a simple but effective scheme to simplify the dendrograms. We descend each dendrogram from the root in a breadth-first fashion and estimate the camera motion for each outgoing edge of the current node. If the camera motion is too small, we collapse the edge, *i.e*. we remove the child and attach its children to the current node. We estimate the camera motion using a scheme similar to [10]: Since each child is in the support set of its parent, we use HOP to compute their overlap region. We then use the change in relative size of the overlap region as an estimate of the zoom and the relative movement of the centroid of the overlap region to estimate the amount of panning or tilt. An example result of this simplification is shown in Fig. 4 (right). We collapse an edge if the size of the overlap region changes by less than 50% and its center shifts by less than 33% of the image size.

4. Experiments

Dataset. To evaluate our algorithm, we selected 34 landmarks that have a large number of details depicted on their Wikipedia pages and downloaded Flickr images in a geographic bounding box around them, resulting in a total of 802,129 images. In two cases (*Westminster Palace/Westminster Abbey* and *Piazza San Marco/Basilica San Marco*), we combined two close-by landmarks for this.

Image Retrieval and Matching. We extract SIFT descriptors from Hessian-Affine interest regions and quantize them using a visual vocabulary of 1M visual words. For each landmark, we build an inverted file index [13, 22]. Images are ranked by $tf * idf$ score and spatially verified by fitting a homography using SCRAMSAC [18]. We reject matches with less than 15 inliers or a $tf * idf$ score below 0.01. Additionally, we reject matches whose homography is close to singular or maps the image borders to a non-convex or self-intersecting polygon.

Results. We apply HIS on the 34 image sets by seeding it once with each image. We then perform dendrogram simplification, which reduces dendrogram size by 55.6% on average. However, the number of covered images decreases by only 0.5%, showing that simplification generally removes redundant Iconoids. HIS typically produces one dendrogram for each facade or view of a landmark and several smaller ones covering individual details. Fig. 5 shows some typical examples of dendrograms produced by HIS. It can be seen that the details discovered at lower bandwidth levels merge into more global structures as β increases. Statistics of the resulting dendrograms are given in Tab. 1. In order to have a unique Iconoid, a support set needs to contain at least 3 images. We therefore only consider Iconoids with support sets of size 3 or higher.

The 34 image sets contain varying amounts of "junk" photos, which were either wrongly geotagged as being close to the landmark, or simply do not depict the landmark. Groups of such photos typically form *singletons*, *i.e*., single Iconoids that are not part of a dendrogram. Another cause of singletons are details with no superordinate structures, *e.g*., paintings in a museum, which explains the high number of singletons in the Louvre.

How many details does HIS find? To answer this question, we rely on Wikipedia authors to provide a list of the relevant details. For each landmark, we downloaded the Wikipedia article in the language version having most details. For the two joint image sets *Westminster Palace and Westminster Abbey* and *Piazza San Marco and Basilica San Marco* we downloaded both Wikipedia articles. We used HTML alt-tags and image captions to obtain labels. We then manually removed unsuitable photos like floor plans or portraits of people associated with the landmark.

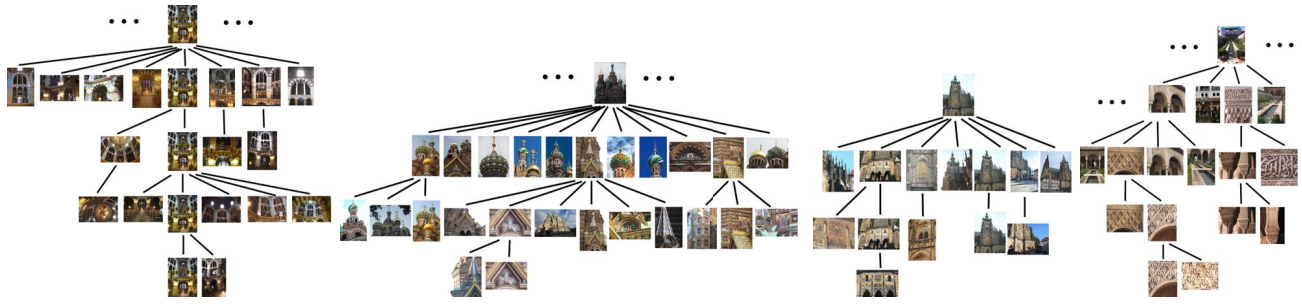
We perform the matching by querying the existing inverted file indices for each landmark with the detail images from Wikipedia. A Wikipedia image matches an Iconoid if they are related by a homography with at least 15 inliers and their homography overlap distance is less than 0.9. To ensure high recall for this matching, we additionally consider matches between Wikipedia details and images directly adjacent to the Iconoid in its matching graph. For these matches, we compute the overlap between the Wikipedia image and the Iconoid via HOP. For each Iconoid, the Wikipedia match with the lowest overlap distance is kept.

There are however two caveats to consider: Firstly, we cannot expect an exhaustive list of details from Wikipedia, and secondly, SIFT-based matching and spatial verification

Landmark	Aachen Cathedral	Alhambra	Angkor Wat	Arc de Triomphe	Bamberg Cathedral	Branden-burg Gate	Chichen Itza	Church on Spilt Blood	Cologne Cathedral	Edinburgh Castle	Florence Cathedral	Hagia Sophia	Louvre	Milan Cathedral	Museée d'Orsay	Nidaros Cathedral	Notre Dame
Images	2,077	55,532	26,485	33,128	964	34,684	9,176	3,588	21,104	27,465	37,035	24,823	69,928	41,192	17,879	581	35,709
Dendrograms	40	562	445	176	22	290	124	21	210	287	296	241	884	341	312	11	266
Mean Dend. Size	7.28	11.22	6.88	12.77	6.86	7.93	8.56	17.33	11.44	11.73	12.82	11.40	8.09	13.75	6.25	4.27	13.42
Singletons	47	563	409	198	16	391	122	20	264	332	289	196	1,046	381	398	11	266
Iconoids	277	5,801	2,850	2,137	128	2,321	1,003	335	2,267	3,128	3,456	2,510	6,974	4,378	1,970	48	3,257
Images Covered	1,192	32,783	13,641	23,731	552	14,221	5,422	2,459	12,900	17,618	28,444	16,911	36,742	25,591	11,775	218	24,841
Wikipedia Details	22	12	5	23	52	9	19	7	18	18	29	26	256	60	36	17	90
WP → Iconoid	10	9	2	14	10	4	7	6	9	12	18	17	99	22	25	1	54

Landmark	Old Town Sq. Prague	Pantheon	Piazza& Basilica San Marco	Plaza de Espana	Reichstag	Sagrada Familia	Sistine Chapel	St Pauls Cathedral	St Peters Basilica	St Stephens Cathedral	St Vitus Cathedral	Taj Mahal	Trevi Fountain	Trinity College	Wat Phra Kaew	Westminster Palace & Abbey	Zwinger
Images	28,349	17,832	40,654	7,689	22,064	17,249	31,337	30,345	32,882	15,399	21,392	11,070	13,502	3,815	29,453	31,010	5,737
Dendrograms	242	105	331	64	181	176	312	200	304	199	215	67	49	72	406	270	59
Mean Dend. Size	10.83	20.41	12.71	8.48	10.59	13.52	11.53	14.32	12.31	9.78	12.20	14.87	15.41	5.33	8.88	12.54	13.71
Singletons	283	95	408	96	224	162	242	268	228	221	130	106	70	75	372	293	49
Iconoids	2,485	1,677	4,009	546	1,866	2,173	3,245	2,735	3,361	1,805	2,324	972	739	373	3,313	3,200	726
Images Covered	19,399	13,960	22,780	2,522	11,013	12,554	25,606	20,367	26,509	8,894	16,569	8,112	10,773	1,621	17,762	22,732	3,814
Wikipedia Details	9	12	36	7	11	37	41	10	52	46	15	37	8	11	13	36	17
WP → Iconoid	7	9	21	3	4	22	14	5	31	19	10	8	6	5	6	16	8
Iconoid Shift [25]																	
Iconoids			411			645	387			606		279		161			218
Images Covered			16,468			10,569	21,038			7,307		6,869		1,286			3,143
WP → Iconoid			13			8	2			14		4		3			6

Table 1: Results of HIS on 34 landmarks. *Images Covered* is the number of images covered by Iconoid support sets.



(a) Aachen Cathedral (b) Church on Spilt Blood (c) Old Town Square Prague (d) Alhambra

Figure 5: Example dendrograms showing detail structures automatically discovered by Hierarchical Iconoid Shift.

using homographies is not suitable for all details. For example, many of the details of *Milan Cathedral* and *Notre Dame* are statues, which were discovered by HIS, but that could not successfully be matched, because they are weakly textured and highly non-planar. Therefore, the numbers we provide should be considered a pessimistic measure of the recall of detail discovery.

The results of this experiment are shown in Tab. 1. *Wikipedia Details* is the number of details depicted in each article and *WP → Iconoid* is the number of Wikipedia images with at least one matching Iconoid. Generally, the number of details discovered depends on the number of images available for the landmark, since a dense enough sampling of the detail space is needed to identify local maxima. HIS therefore finds a large fraction of the details of *Piazza & Basilica San Marco* and *Notre Dame*, but almost none for *Nidaros Cathedral*, which has only 581 images.

Some examples of Iconoids with matching Wikipedia details are shown in Fig. 6. The Wikipedia image captions can serve as very accurate labels for the Iconoids. Such high-quality labels cannot be found by simply examining frequently occurring terms in photo titles and tags [16, 21, 27], since photographers often do not make an effort to correctly label each landmark detail. Overall, about half of the Wikipedia details have a matching Iconoid. Fig. 7 illustrates main causes of missed Wikipedia details.

How does HIS compare to Iconoid Shift? We compare HIS to Iconoid Shift [25] on a subset of 7 landmarks (lower block of Tab. 1). The Iconoid Shift clustering covers almost the same number of images, but has a much coarser granularity, resulting in significantly less Iconoids. The reason for this is that IS performs mode search only on a single fixed scale ($\beta = 0.9$ in our experiment), while HIS finds modes on all scales. Comparing the number of discovered

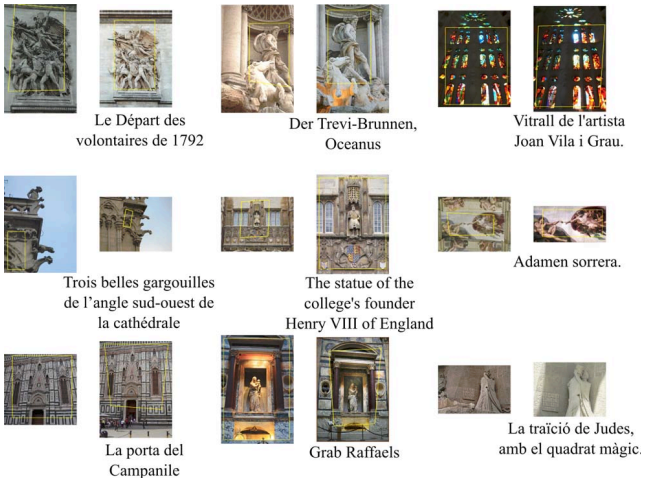


Figure 6: Example Iconoids (left) with matched Wikipedia images and labels (right).

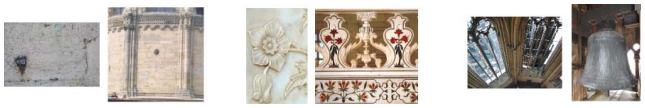


Figure 7: Objects from Wikipedia not discovered by HIS. Left: Objects are historically relevant, but not visually striking. Middle: Objects are hard to photograph in isolation without a telephoto lens or cropping them out. For such objects, only the superordinate structure will form a cluster. Right: Objects are hard to access or not always open to the public.

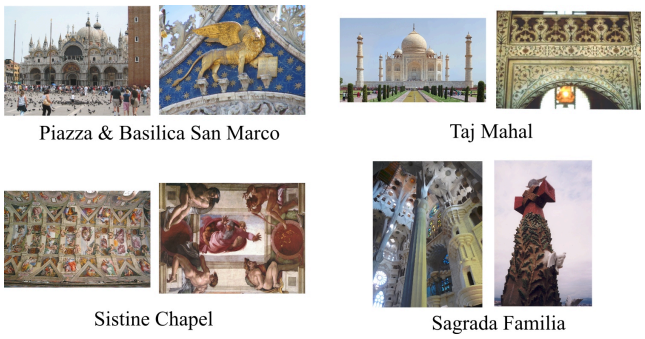


Figure 8: Objects discovered by both HIS and IS [25] (left) vs. objects discovered only by HIS (right). While both algorithms find larger-scale structures, only HIS finds also finds small details.

Wikipedia details, it can be seen that especially for landmarks like *Sagrada Familia* or the *Sistine Chapel* that have many close-by details, IS finds significantly less details. On the other hand, for landmarks like *Zwinger* where the objects on Wikipedia are mostly non-hierarchical, e.g. several distinct buildings, the advantage of HIS is smaller. Fig. 8 compares objects discovered by both algorithms (left) to objects only discovered by HIS.

Applications. In our impression, many of the details that do not have a match in Wikipedia would likely be worth

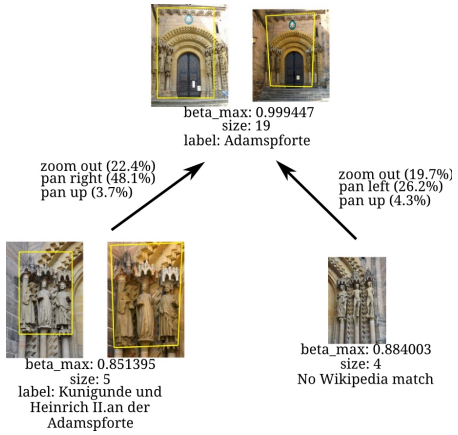


Figure 9: Finding new details to add to Wikipedia. The right child of the node showing the gate is not present on Wikipedia, but we can propose a place to insert it into the article based on its parent and sibling in the dendrogram and the estimated camera motion between them.

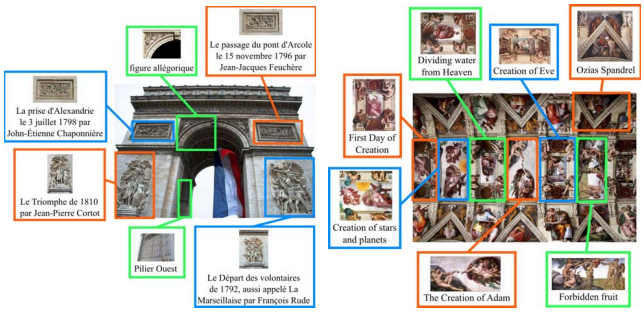


Figure 10: Discovered details of the Arc de Triomphe and the Sistine Chapel ceiling.

mentioning in the article. A potential application of HIS would therefore be to suggest new details to be added to Wikipedia. We use the structure of the HTML heading tags to build a tree of each article and insert each image as a leaf under its respective heading. We then match the Wikipedia images against the discovered Iconoids to create links between the Wikipedia graphs and the Hierarchical Iconoid Shift dendrograms. By exploiting links between the HIS dendrograms and the Wikipedia article tree, it is possible to guess the article section a new detail should be added to. As an example, Fig. 9 shows a part of a dendrogram where a figure group in the left part of a gate is present in Wikipedia while the figure group in the right part of the gate is missing. From the adjacent nodes in the dendrogram and the camera motion estimation described above, we know the scene parts that the figure group is related to and where it is located relative to them. In this case, we know that the right figure group is a part of the gate since the gate is its parent in the dendrogram and the camera move associated with the edge is "zoom out". We can therefore propose to add this node to the article section describing the main gate.

Another application is depicted in Fig. 1 and Fig. 10. Using the discovered dendrograms describing the scene structure, as well as the image clusters associated with each node, we can build visualizations of landmark building facades that are far more useful to the user of a mobile visual search application than the name of the main landmark alone. The images in the Iconoid clusters can serve two purposes here. Firstly, using HOP, we can precisely localize even very small details on a large facade since the images in the clusters can help bridge larger scale changes than normal local feature matching can handle. Secondly, as in [8], the clusters provide additional images for each detail showing it under different lighting conditions and viewing angles and can thus be used as an "offline query expansion" to make recognition of these details more robust compared to matching against the Wikipedia images alone.

5. Conclusion

In this work, we have proposed *Hierarchical Medoid Shift*, a new variant of Medoid Shift [19] that, instead of a flat clustering at a single scale, produces a dendrogram of clusters by continuously sweeping over all scales. HMS has the same complexity as regular Medoid Shift, is completely parameter-free and easy to parallelize. Based on HMS, we have proposed *Hierarchical Iconoid Shift*, a new algorithm for clustering internet photo collections that produces a hierarchy of clusters that includes both small details as well as full views of a landmark's facade. We have demonstrated our algorithm on a set of 34 landmarks with a rich detail structure and shown that it discovers many of the details depicted on Wikipedia and significantly outperforms Iconoid Shift [25] in terms of the number of discovered details. The scene hierarchies produced by HIS could be useful for a large range of applications including landmark description and visual recognition of detail structures.

Acknowledgments. We gratefully acknowledge support by the DFG Excellence Cluster UMIC (DFG EXC 89) and by a Google Faculty Research Award.

References

[1] Y. Avrithis, Y. Kalantidis, G. Tolias, and E. Spyrou. Retrieving Landmark and Non-Landmark Images from Community Photo Collections. In *ACM Multimedia*, 2010.

[2] F. Chin and D. Houck. Algorithms for Updating Minimal Spanning Trees. *J. Computer and System Sciences*, 16(3):333–344, 1978.

[3] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *PAMI*, 24(5):603–619, 2002.

[4] D. Comaniciu, V. Ramesh, and P. Meer. The Variable Bandwidth Mean Shift and Data-Driven Scale Selection. In *ICCV*, 2001.

[5] D. Crandall, L. Backstrom, D. Huttenlocher, and J. Kleinberg. Mapping the World's Photos. In *WWW*, 2009.

[6] D. DeMenthon. Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis. In *SMVP*, 2002.

[7] B. Epshtein, E. Ofek, Y. Wexler, and P. Zhang. Hierarchical Photo Organization using Geo-Relevance. In *ACM GIS*, 2007.

[8] S. Gammeter, T. Quack, D. Tingdahl, and L. Van Gool. Size Does Matter: Improving Object Recognition and 3D Reconstruction with Cross-Media Analysis of Image Clusters. In *ECCV*, 2010.

[9] S. Gammeter, T. Quack, and L. Van Gool. I Know What You Did Last Summer: Object-Level Auto-Annotation of Holiday Snaps. In *ICCV*, 2009.

[10] A. Ladikos, E. Boyer, N. Navab, and S. Ilic. Region graphs for organizing image collections. In *RMLE*, 2010.

[11] Y. Leung, J.-S. Zhang, and Z.-B. Xu. Clustering by Scale-Space Filtering. *PAMI*, 22(12):1396–1410, 2000.

[12] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In *ECCV*, 2008.

[13] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *CVPR*, 2007.

[14] J. Philbin, J. Sivic, and A. Zisserman. Geometric Latent Dirichlet Allocation on a Matching Graph for Large-scale Image Datasets. *IJCV*, 95:138–153, 2011.

[15] J. Philbin and A. Zisserman. Object Mining using a Matching Graph on Very Large Image Collections. In *ICCVGIP*, 2008.

[16] T. Quack, B. Leibe, and L. Van Gool. World-Scale Mining of Objects and Events from Community Photo Collections. In *CIVR*, 2008.

[17] B. C. Russell, R. Martin-Brualla, D. J. Butler, S. M. Seitz, and L. Zettlemoyer. 3D Wikipedia: Using Online Text to Automatically Label and Navigate Reconstructed Geometry. *ACM TOG*, 32(6), November 2013.

[18] T. Sattler, B. Leibe, and L. Kobbelt. SCRAMSAC: Improving RANSAC's Efficiency with a Spatial Consistency Filter. In *ICCV*, 2009.

[19] Y. Sheikh, E. Khan, and T. Kanade. Mode-Seeking by Medoidshifts. In *CVPR*, 2007.

[20] I. Simon and S. Seitz. Scene Segmentation Using the Wisdom of Crowds. In *ECCV*, 2008.

[21] I. Simon, N. Snavely, and S. Seitz. Scene Summarization for Online Image Collections. In *ICCV*, 2007.

[22] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *ICCV*, 2003.

[23] N. Snavely, S. Seitz, and R. Szeliski. Photo Tourism: Exploring Photo Collections in 3D. In *SIGGRAPH*, 2006.

[24] P. Vatturi and W.-K. Wong. Category Detection using Hierarchical Mean Shift. In *KDD*, 2009.

[25] T. Weyand and B. Leibe. Discovering Favorite Views of Popular Places with Iconoid Shift. In *ICCV*, 2011.

[26] A. P. Witkin. Scale-space filtering: A new approach to multi-scale description. In *ICASSP*, 1984.

[27] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a Web-Scale Landmark Recognition Engine. In *CVPR*, 2009.