Qualitative Representation of Images

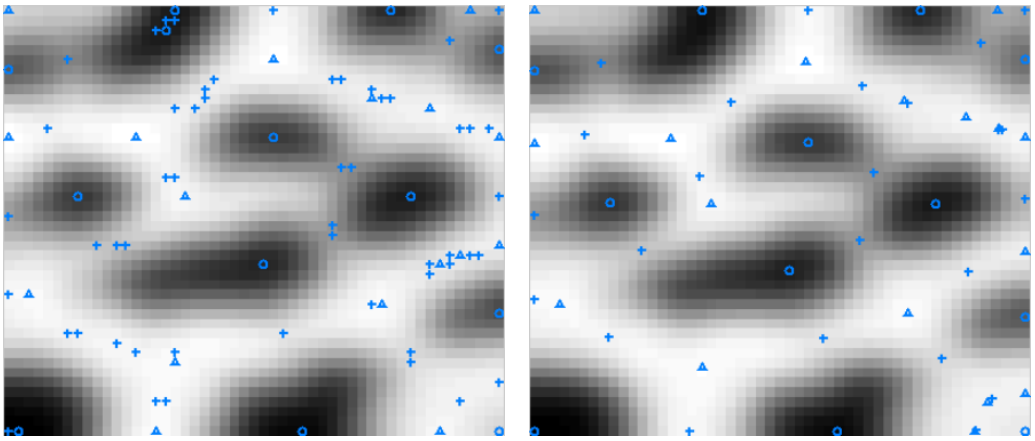
Firat Kalaycilar and Benjamin Kimia

September 1, 2013

Contents

1 TODO List

1. ~~Include experimental results (copy them from the tech report ppt)~~
2. ~~Add visual examples of the issues of the current implementation~~
3. ~~Algorithm correction: no need for Mask images, Label images are sufficient~~
4. Integrate Prof. Kimia's comments
5. ~~Add examples for each event pixel classification case~~
6. Add more questions and answers
7. Clean up the code



(a) 8 neighbor local detection (b) Our algorithm

Figure 1: Comparison of critical points computed using different algorithms. Observe the differences between the saddle points (blue pluses).

2 Introduction

The goal of this project is to develop a generative appearance model to be used in several computer vision applications such as image representation, image generation, image matching, multiview image calibration *etc.* Our appearance model is called topological appearance graph (TAG) whose design is inspired from the idea of sketching functions using their critical points. The TAG is a graph whose nodes are the image critical points (minimum, maximum and saddle points) and the edges between them are the topographic curves connecting them.

The idea of using a critical point graph is not novel. The most well known representations are Surface Networks [20, 21], Critical Point Configuration Graphs [19] and Morse-Smale Complexes [2]. These are actually nearly identical to TAG, but have not been used in computer vision applications extensively. Our plan is to augment the TAG by integrating the edges and curves of an image into the representation. However, this document will not discuss the augmented version, since we have not implemented anything related, yet.

Correct critical point detection is essential for this project. But, it is a challenging task, especially the detection of saddle points. Local saddle detectors produce many false positives and false negatives. Therefore, we use a different definition of saddle points when designing our algorithms. We define saddle points as the intersection points of watershed and watercourse lines. See Figure 1 for a comparison of our method to the traditional 8 neighbor local detection.

There are mainly two contributions in this project.

1. Saddle detection as intersection of watershed and watercourse lines. While traditional approaches use local processes, this proposed approach is global. (probably

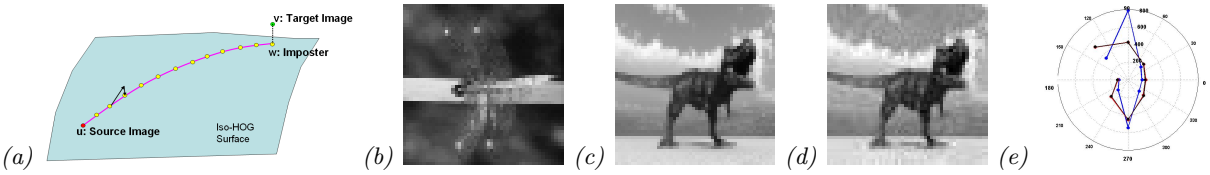


Figure 2: The equivalent classes of images sharing the same HOG representation is huge, to the point that any image patch (b) (HOG in red) can be deformed toward any other image patch (c) (HOG in blue), while maintaining the same HOG representation, thus creating an imposter (d) (HOG in black), with minor differences from the target image.

novel)

(a) Detection of extended minima/maxima/saddle as an integral part of the algorithm. (not so sure if new)

2. Use of critical points and surface networks in various computer vision tasks such as image matching, multiview calibration, image generation *etc.*

For more details on the goals of this project, please read what was proposed in the NSF'12 proposal:

A Novel Approach to Modeling Appearance: Popular HOG/SIFT-like models capture local appearance as a histogram of gradient orientations. The use of gradients and orientation direction factor out additive and multiplicative changes, respectively, while the use of a histogram builds resilience against small deformations. Together, they build some invariance against illumination, pose and other changes and these descriptors have been very effective in recognition systems. Unfortunately, the equivalence class of images mapping to a collection of such histograms, either at sparsely or densely sampled points, is huge and uncontrolled, Figure 2. Such defects cannot be fully observed at the current scale of datasets, but are expected to become prominent as the variability and number of exemplars and categories both increase significantly.

We seek a generative representation where the induced equivalence class of images is better modeled. Elementary Calculus prescribes sketching a function of one variable using its critical points, thus inducing an intuitive equivalence class for a set of specified critical points. What is the two-dimensional analogue for sketching an image when considered as a two-dimensional scalar field? Is the resulting qualitative structure immune to changes induced by visual transformations? Cayley, Maxwell, and mathematically minded geographers after them have studied the qualitative topographical structure of a landscape based on a surface network of topographical curves connecting critical points of image gradient, namely, peaks, pits and passes [3, 18, 20, 19, 25, 38, 21]. In this proposal we refer to the graph whose nodes are critical points and whose links are topographical curves as the **Topological Appearance Graph (TAG)**.

We will not cover the history of this area, which is complex, involving confusion between local and global definitions [27, 7, 11, 15, 24] and an evolving notion of genericity [13, 5, 6]. We adopt Jordan's definition [11] with a weak genericity condition [5], as in

[22, 23, 24]; see [12]. Beyond geography, the topographical graphs have also been used in computer vision, but with a rather sparse history [19, 26, 25, 32, 4, 30, 29]. Note that [9] uses a graph where stable critical points of the Laplacian image are connected by region grouping. The excellent image matching results encourage our approach which in contrast is based on critical points of the image and is connected by topographic curves.

The insight underlying the proposed use of a TAG for capturing appearance is that the location and value of critical points are both invariant to additive and multiplicative image changes and the topological neighborhood structure is invariant to a range of deformations. Thus, critical points can potentially serve as a replacement for both *(i)* interest points and *(ii)* regional descriptors. We probed the repeatability of critical points using dense correspondence in a multiview, multi-illumination dataset [33, 34]: the top 10% most significant (ranked by curvature) critical points while roughly equal in number, outperform both Harris and SIFT in repeatability, both in narrow and wide base-line conditions, and under illumination changes. Thus, they can potentially be used together with the local graph structure as a descriptor in a traditional recognition approach. We will implement this approach as a baseline so as to enable progress in the remaining components of the proposal, while a more intriguing approach to capturing appearance is developed below.

Robust Computation Using a Local Form Catalogue: We aim to resolve ambiguity in the local, discrete domain computation of critical points, mainly occurring at saddle points [8], by ensuring local form is one of the legal configurations [19]. Second, numerical instabilities generically arise when nearly parallel curves falling below resolution merge, thus producing illegal graph connectivity [19, 26, 25, 30, 29]. We posit that the main reason is that the image grid is used to compute topographic curves. We aim to develop a pixel-independent computational domain where multiple curves can simultaneously occupy a pixel and instead focus on satisfying the local form. The main impetus in achieving numerical robustness is that the computed graph should be the exact representation of an approximation of the image, as opposed to an approximation of the exact representation.

Patch Similarity via Edit Distance: Descriptors are primarily used for measuring similarities between two patches as a basis of recognition. In the case of HOG, the similarity is extrinsic, *i.e.*, the Euclidean distance in the representation space. It is much more appropriate, however, to measure similarities intrinsically by measuring the extent of deformation needed to morph one patch to another. Unfortunately, the search for geodesics in the appearance space is combinatorially impossible. Observe, however, that deformation paths can be represented by a discrete combination of critical point transitions (where one appearance equivalence class changes into another), thus reducing the underlying dimensionality, Figure 3. This is analogous to the approach of measuring the similarity between two shapes using the transitions of the medial axis where the evolution of the hair-like instabilities are themselves the representation of a shape deformation path. We plan to develop an edit distance metric between two TAGs based on the transition of the TAG.

TAG Simplification via Structural Smoothing and Self Similarity: The TAG

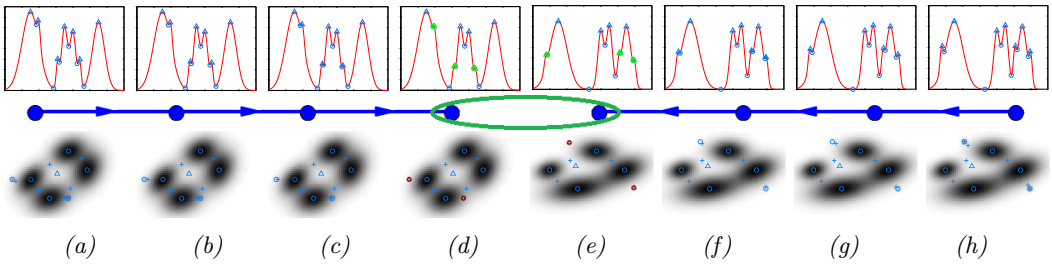
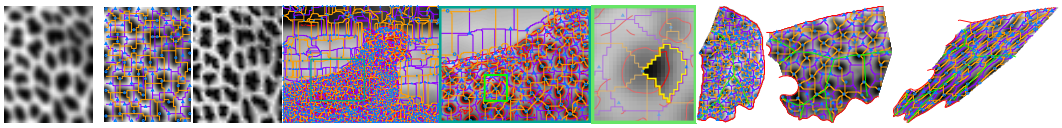


Figure 3: (First row) Observe in (a) and (h) two structurally similar 1D functions which have different critical points due to small perturbations. A sketch of the geodesic path represented as a pair of deformation paths, one from each shape, simplifies each function in discrete steps, as max and min annihilate, until the two functions are topologically equivalent. The second row shows a 2D analogue.

model partitions an image into slope districts. These are monotonic patches and can be qualitatively reconstructed by interpolating intensities at graph nodes and links, Figure 4(a-c). Additional accuracy will result if continuity across slope district boundaries are enforced (not shown here). A key question is whether the representation description length can be further reduced without adversely affecting its generative power, thus producing more effective equivalent classes. We plan to investigate three directions to address this point. First, we plan to iteratively effect a transition starting from the least-cost pair of critical points which are annihilated [2, 35] and the process iterated up to some total cost. This type of smoothing is structural/surgical and local, in contrast to diffusion-based models which remove fine detail at the cost of degrading coarse structure. Second, we will cross-correlate a local patch of the appearance graph with the rest of the graph using the above proposed edit distance measure to find self-similar structure [28]. A clustering of self-similar patches can lead to the notion of a *TAG-texton*. Since TAG is topological, affine transformations and undulations of the surface, like those on the leopard in Figure 4 should leave the graph intact. Note that local parts of the graphs need not have identical topology as long as the edit cost between them is low. Finally, for fragments whose TAG description length remains high after the above steps, we resort to a statistical model capturing the distribution of node degree, node intensities, link distances, and perhaps second-order attributes so that textures can be effectively synthesized using the TAG grammar. These three steps should handle shaded areas (faces), regular and irregular structural textures (carpet, bricks, and pebbles), near stochastic (wood grain) and stochastic textures (sand) [16].

3 Related Work

The included NSF proposal section is a good guide for the related work. For more details, please read Prof. Kimia's tech report with the title "On Topographic Curves in Images".



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 4: A small texture patch (a) with the Topographical Appearance Graph (TAG) superimposed in (b) is reconstructed in (c) using diffusion from TAG-stored intensities. (d) The TAG of a full image where peaks and pits are triangle and circulars icons, respectively, watersheds/watercourses are shown in yellow/purple curves, respectively, and saddle points are their intersection points. Red curves are contour fragments. (f) The intersection region of hills and dales is a slope district which is a monotonic patch of the image which can be effectively represented just from the intensity values at the nodes. (g) The "leopard arm" fragment with its TAG. (h) The "leopard torso" fragment with its TAG (i) The same torso fragment and its TAG after an affine transformation.

4 Theoretical Aspect

4.1 Terminology and Notation

Concept	Definition	Notation
Topographical surface		$z = f(x, y)$
Gradient	$\nabla f = [f_x, f_y]$	∇f
Ortho-gradient	$\nabla f^{\perp} = [-f_y, f_x]$	$\nabla f^{\perp}$
Hessian	$H = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}$	H
Determinant of Hessian	$\lvert H\rvert = \det(H) = f_{xx}f_{yy} - f_{xy}^2$	$\lvert H\rvert, \det(H)$
Trace of Hessian	$\mathrm{Tr}(H) = f_{xx} + f_{yy}$	$\mathrm{Tr}(H)$
Laplacian	$\Delta f = f_{xx} + f_{yy} = f_{uu} + f_{vv}$	Δf
Eccentricity	$\varepsilon^2 = \frac{1}{4}(f_{xx} - f_{yy})^2 + f_{xy}^2$	ε
Principal curvatures	Eigenvalues of H, $\Delta f \pm \varepsilon$	λ_1, λ_2
Principal curvature directions	Eigenvectors of H	e_1, e_2
Regular point	Non-critical point where $\lvert\nabla f\rvert > 0$	
Peak (Morse max)	$\lvert\nabla f\rvert = 0$, $\det(H) > 0$, $\mathrm{Tr}(H) < 0$	$\triangle$
Pit (Morse min)	$\lvert\nabla f\rvert = 0$, $\det(H) > 0$, $\mathrm{Tr}(H) > 0$	$\circ$
Pass/bar (Morse saddle)	$\nabla f = 0$, $\det(H) < 0$	$+$
Degenerate critical point	$\det(H) = 0$	
Level-set contour	$\beta(0) = P$, $\beta'(s) = -\nabla f^{\perp}(\beta(s))/\lvert\nabla f^{\perp}(\beta(s))\rvert$	$\beta(s)$
Slope line (integral curve)	$\alpha(0) = P$, $\alpha'(s) = -\nabla f(\alpha(s))/\lvert\nabla f(\alpha(s))\rvert$	$\alpha(s)$
Slope line segment	The slope line between two adjacent critical points.	
Hill associated with peak p_0	$\{p \mid \exists$ slope line segment through p that ends at $p_0\}$	$\mathrm{Hill}(p_0)$
Dale associated with pit p_0	$\{p \mid \exists$ slope line segment through p that ends at $p_0\}$	$\mathrm{Dale}(p_0)$
Watershed line	Dale boundaries. Special slope lines connecting saddles to peaks. According to Rothe/Reiger [24], this definition may not be true under degenerate cases.	WS
Watercourse line	Hill boundaries. Special slope lines connecting saddles to pits. According to Rothe/Reiger [24], this definition may not be true under degenerate cases.	WC
Slope district	Non-empty intersection of a hill and a dale [19]; monotonic region.	SD

Notation 1 *The **gradient** and **Hessian** of a function $f(x, y)$ are denoted by $\nabla f = [f_x, f_y]$ and $H = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}$.*

Definition 1 *A **non-degenerate critical point** of a function $f(x, y)$ occur at points*

where $|\nabla f| = 0$, *and* $|H_f| \neq 0$.

Notation 2 *Recall that the derivative of* f *in the direction of the unit vector* T *which is parametrized by variable* ξ *is*

$$\frac{\partial f}{\partial \xi} = \nabla f \cdot T.$$

4.2 Questions, Propositions and Proofs

The slope line at each point can be obtained by integrating an ODE $\alpha'(s) = -\nabla f(\alpha(s))$ where $\alpha(s)$ is the slope line.

Proposition 1 *There is only one slope line through each regular point* P *since the ODE,* $\alpha'(s) = -\nabla f(\alpha(s))$, *with initial value for* $\alpha(0) = P$ *has a unique solution.*

This is an explicit ODE for which there is a unique solution by the existense and uniqueness theorem [24]. Also see [36].

Proposition 2 *The end points of a slope line segment, i.e., the slope line between two consecutive critical points, cannot both be a pit or both be a peak.*

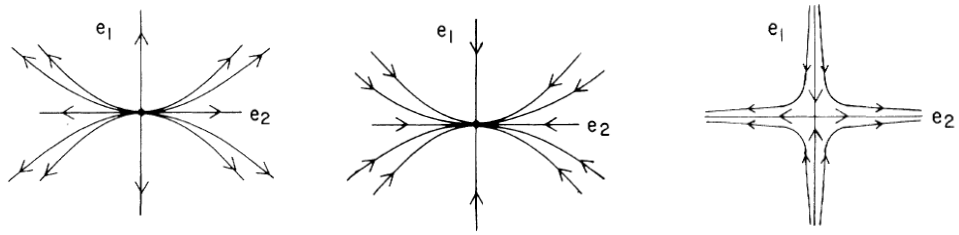
Let both p_1 and p_2 be two consecutive peaks on a slope line segment. Without loss of generality, assume that the downward flow is from p_1 to p_2. Since all slope lines around a peak must be outgoing, there is a contradiction in p_2 having an incoming slope line. Therefore, we can say that both end points cannot be a peak. A similar proof can be given for the pit case.

Corollary 1 *Can we have two slope lines meeting tangentially or intersecting transversely at a regular point?*

No. Assume two slope lines meet at regular point p. Then, if we solve the slope line ODE with $\alpha(0) = p$, we should get two separate curves. But, we already know that the solution is unique, so that would be a contradiction.

Question 1 *How do slope lines behave near min/max points?*

All slope lines approach mininima/maxima tangent to one of the principal curvatures [19]. See Figures 5a and 5b. If the point is umbilical, the slope lines approach from all directions. This can be verified using a Monge patch. Let $f(x, y) = ax^2 + by^2$ be the surface of interest where the signs of a and b are the same. Note that $(0, 0)$ is either a max or a min point. Let $p_0 = (\epsilon \cos\theta, \epsilon \sin\theta)$ be a point on a slope line originating from $(0, 0)$. Observe that as ϵ goes to 0, the vector $p_0 - (0, 0) = (\epsilon \cos\theta, \epsilon \sin\theta)$ becomes tangent to $\nabla f(p_0) = (2a\epsilon \cos\theta, 2b\epsilon \sin\theta)$. This implies that $(2a\epsilon \cos\theta, 2b\epsilon \sin\theta) \cdot (\epsilon \sin\theta, -\epsilon \cos\theta) = 0$ which reduces to $(a - b) \sin 2\theta = 0$. If the point is not umbilical, $a \neq b$, then $\theta = 0$ or $\theta = \pi/2$ which are the pricincipal curvature directions. On the other hand, if $a = b$, the value of θ is arbitrary.



(a) max (b) min (c) saddle

Figure 5: Slope line behaviour near critical points. e_1 and e_2 are the principal curvature directions. Figures were taken from [19].

Question 2 *How do slope lines behave near saddle points?*

See Figure 5c. We can use the same argument to show that the slope lines approach saddles along e_1 and e_2. Consider the Monge patch $f(x, y) = ax^2 + by^2$ where a and b have opposite signs. Using the same approach we applied in the previous question, we obtain $(a - b) \sin 2\theta = 0$. In this case, it is guaranteed that $a \neq b$. Thus, $\theta = 0$ or $\theta = \pi/2$ are the solutions which correspond to the principle curvature directions.

Question 3 *How do we characterize the hill corresponding to each peak?*

Let p_0 be a peak. Then, $\text{Hill}(p_0) = \{p \mid \exists$ slope line segment through p whose one end point is $p_0\}$. Similarly, if p_0 is a pit, $\text{Dale}(p_0) = \{p \mid \exists$ slope line segment through p whose one end point is $p_0\}$.

Question 4 *Can two saddle points be connected with watershed/watercourse lines in generic images?*

This can happen, but maybe an intrinsically unstable event [25], so it may not be generic. We have experience that saddle-saddle connections act like a watershed and watercourse line at the same time.

Question 5 *Can hills/dales be multiply connected?*

No. Suppose we have a hill with two disconnected regions. The points in the region without the peak must be connected to the peak with slope lines by definition. Since a region cannot be disconnected from a region containing the peak and have slope lines connected to the peak at the same time, we can say that hills cannot be composed of disconnected regions. [pathwise connectivity]

Question 6 *Do hills partition the image domain completely? In other words, is there any point in the domain that does not belong to any hill?*

Every image point either belongs to a hill (corresponding to a peak) or to a watercourse line (which separates two hills). Similarly, every image point belongs to a dale (which corresponds to a pit) or to a watershed line (which separates two dales).

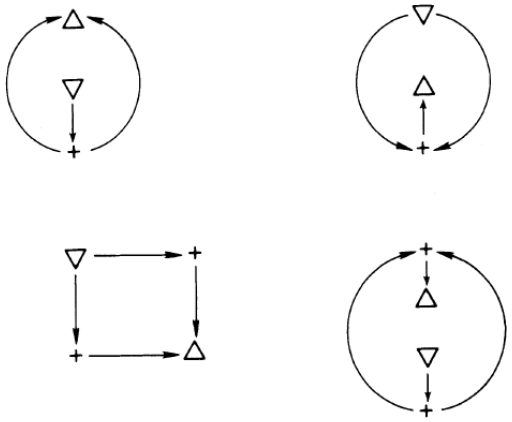


Figure 6: Catalog of slope districts. Taken from [19].

Proposition 3 *Hills are tightly surrounded by watercourse lines. In other words, watercourse lines correspond to hill boundaries.*

Yes. Don't know how to prove this!

Question 7 *Are saddle-saddle connections possible?*

Yes. See Nackman's slope district catalog [19]. (Figure 6). It is not clear whether watershed, watercourse or both simultaneously connect them.

Question 8 *Are saddle-saddle connections generic?*

TODO

Question 9 *Is there any critical point that does not belong to a watershed/watercourse line?*

No. There are slope line segments going in or out of each critical point. And those segments always have another end point (possibly out of boundaries in the case of images) which is also a critical point. Based on the type of the end points, these slope line segments are either watershed or watercourse lines.

Question 10 *Can there be any watershed/watercourse line without any critical points?*

No. Watershed/watercourse lines are slope line segments, so by definition their end points are always critical points.

Proposition 4 *There is no other max point on a hill boundary.*

Assume that p is a max point on the boundary of hill h. In some neighborhood of p with no other critical points, all the upward flowing slope lines converge to p. Since this neighborhood intersects h, all the points in the intersection must belong to h and the

hill corresponding to p at the same time. Since this is not possible, we can say that there cannot be any peaks on a hill boundary.
Another explanation: Hill boundaries are composed of watercourse lines, so there can only be regular, min and saddle points.

Proposition 5 *There is at least one saddle and one min (max) point on a hill (dale) boundary.*

[This proof ignores the saddle-saddle connections. INCORRECT.] Since hills are bounded by watercourse lines whose end points are always a min and a saddle point, it is clear that there must be at least one min and one saddle point on a hill boundary.

Proposition 6 *Saddles and min points are arranged in an alternating pattern on a hill boundary.*

[Probably INCORRECT] Since there is no slope line with min-min or saddle-saddle end points, min and saddle points must be in an alternating pattern.

Proposition 7 *The number of saddles and min points is the same on a hill boundary.*

[Probably INCORRECT] The closed alternating pattern guarantees that the number of saddles and min points is the same.

Question 11 *Can there be another critical point inside a hill (dale) other than the associated peak (pit)?*

There is an example showing a special case in [2] on page 3. A min and a saddle point are inside the centeral hill. This example is important, because it is generic and might affect our definition of slope districts.

Corollary 2 *Slope lines from a hill (dale) meet the hill boundary transversely at a saddle and tangentially at a min.*

Figure 5 gives an idea about why this should be correct. Observe how slope lines meet the critical points.

4.3 Ridge and Valley Topographic Curves

Definition 2 *A* ridge *or* valley *point of $z = f(x, y)$ is a point where*

$$\overrightarrow{\nabla f}.\overrightarrow{\nu_1} = 0, \tag{1}$$

where ν_1 is the eigenvector with the largest eigenvalue of the Hessian of f, and where the eigenvalue $\lambda_1 > 0$ or $\lambda_1 < 0$, respectively. [Letting ξ parametrize direction $\overrightarrow{\nu}$, we have]

In analogy to the condition in 1D used to distinguish minima from maxima among the extrema of 1D functions, namely, whether $f''(x)$ is positive or negative, respectively, ridges and valleys are distinguished using the second derivative in the direction ν_1,

$$\frac{\partial^2 f}{\partial \xi^2} = \nu_1^T H \nu_1 = \nu_1^T (\lambda_1 \nu_1) = \lambda_1,$$

which gives just the eigenvalue λ_1. Thus, if $\lambda_1 < 0$, we have a ridge and if $\lambda_1 > 0$ we have a valley. The case $\lambda_1 = 0$ also implies $\lambda_2 = 0$ since λ_1 is the larger eigenvalue. In such a case higher-order derivatives needs to be consulted to determine a ridge.

5 Computing Critical Points and Their Connectivity

This section presents two important algorithms required to compute the critical points of an image and their connectivity. The first algorithm detects both the minima and watershed pixels and is called $min\_watershed\_detector(I, \sigma)$ where I is the image and σ is a smoothing parameter. Note that the same function can be used to detect maxima and watercourse by inverting the image. Its implementation is very similar to the standard watershed transform which works as follows [37]: A set of seed pixels is selected and given different labels. The pixels surrounding the seed pixels are inserted into a priority queue where the pixel priority is based on how low its intensity value is. Then, the pixel with highest priority is chosen. If its already labeled neighbors have the same label, it is given that label. Otherwise, it is kept unlabeled. Then, all unlabeled neighboring pixels are added to the priority queue. The algorithm goes on like this until the priority queue becomes empty. At the end, the labeled pixels are the basins while the unlabeled ones correspond to watershed pixels. The key differences of our algorithm from the traditional one are: *(i)* There is no need for an apriori detection of minima (the seed pixels) since the detection is an integral part of the algorithm; *(ii)* The algorithm keeps track of the identities of the minima giving rise to each watershed pixel. Note that $min\_watershed\_detector(I, \sigma)$ may also be implemented by separating the detection of extended minima[1] as a first, separate stage, and then using the traditional watershed transform to label pixels either as the watershed pixels or basins. Next, using the basin labels that are neighbors to each watershed pixel, the identities of the minima giving rise to each watershed pixel can be found[2].

The second algorithm, $TAG\_detector(I, \sigma)$, aims to find saddles and then connect all the critical points, namely, minima, maxima and saddle points in a graph structure called Topological Appearance Graph (TAG). It uses the minima and watershed pixels detected by the $min\_watershed\_detector(I, \sigma)$ and the maxima and watercourse pixels detcted by the $min\_watershed\_detector(-I, \sigma)$. Since our definition of saddles is the intersection of watershed and watercourse lines, it utilizes the watershed/watercourse pixels to localize the saddles. In addition, since the ids of minima/maxima giving rise to

[1] An extended minima is a connected cluster of pixels with the same intensity value surrounded by higher intensity pixels. An implementation of this is MATLAB's *imregionalmin*.

[2] Not sure if this would be identical to what we compute in our algorithm.

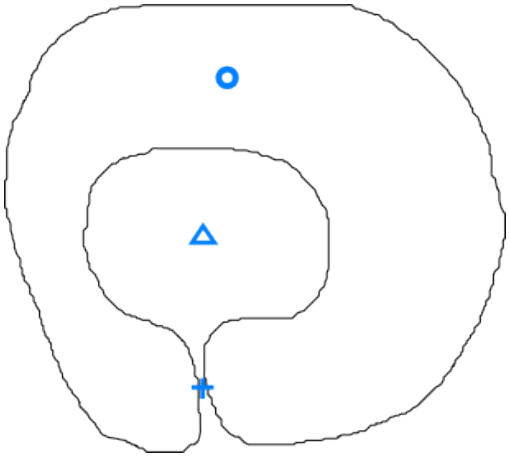


Figure 7: A self-colliding basin. This case is currently ignored by the algorithm.

watershed/watercourse pixels are known, the algorithm is able to establish connections between all the critical points which results in the TAG of the input image I.

function $min\_watershed\_detector(I, \sigma)$**:**

Summary: The goal is to find all the minima and watershed pixels with the ids of the basins (minima) who gave rise to them. In order to achieve this, the intensity values in the image are first sorted from low to high. As in the traditional watershed transform, the intensity is thought of as rising water levels. Then, we visit each water level in increasing order and find out which pixels have just been covered with water (we call them *event pixels*). Next, we classify each connected component of the event pixels in one of three classes: *(i)* minimum, *(ii)* watershed pixel or *(iii)* basin growth (all pixels of a connected component are given the same label[3]). Note that if the connected component has more than one pixel, the result is an *extended* minimum, *extended* watershed or *extended* basin growth. Once all the water levels are visited, the algorithm stops. For the detailed algorithm steps, see below:

Failure mode: Self-collisions are not handled (Figure 7). Some watershed pixels may not be detected which will cause the algorithm to miss the saddles due to self-collisions.

Input:

- Image I
- σ for smoothing

[3]Prof. Kimia asked me if it is possible to assign the pixels of a connected component to different classes. For example, can some part of the connected component be a basin growth while the other parts are watershed pixels? I do not know the answer. We need an explanation.

Output:

- ws: the unorganized list of watershed pixels
- $mins$: the minima of I
- ws_basins: ids of the minima giving rise to each watershed pixel

Local variables:

- i: the index of the loop visiting the sorted water levels.
- $sorted\_I$: the sorted array of unique intensity values in I.
- $Water\_Level$: the level of the rising water.
- $Label\_prev$: an image with the same size as I. Each pixel holds the basin id assigned to it before visiting the current water level. 0 means "not assigned" yet.
- $Label\_curr$: an image with the same size as I. Each pixel holds the basin id assigned to it after visiting the current water level. 0 means "not assigned" yet.
- num_min: the number of minima.

Algorithm:

1. Pad the image boundary using the mirror reflection of the image. I use $(7\sigma - 1)/2$ pixels for padding each side. The motivation for using mirror reflection instead of constant value continuation is that mirror reflection causes most of the boundary pixels to become either critical points or watershed/watercourse pixels. This is violated only when there is a direct transition from a minimum to maximum along the boundary, *i.e.*the absence of a saddle point on the boundary to form a watershed/watercourse line. See Figure 8.
2. Smooth the padded image with a Gaussian with standard deviation σ. [4]
3. Sort all the intensity values in the image in ascending order to create $sorted\_I$.
4. Start a loop with $i = 1$, $Label\_prev$ = initially all 0's and $num\_min = 0$
   i. Set $Water\_Level = sorted\_I[i]$.
   ii. Compute the mask $Mask\_curr$: If the image intensity $\leq Water\_Level$, the mask is set to -1. Otherwise, it is set 0. The reason for using -1 instead of 1 will become clear shortly.
   iii. $Label\_curr = Mask\_curr$.

[4] This step may not be required in the future, as structural smoothing after graph construction may be more meaningful.

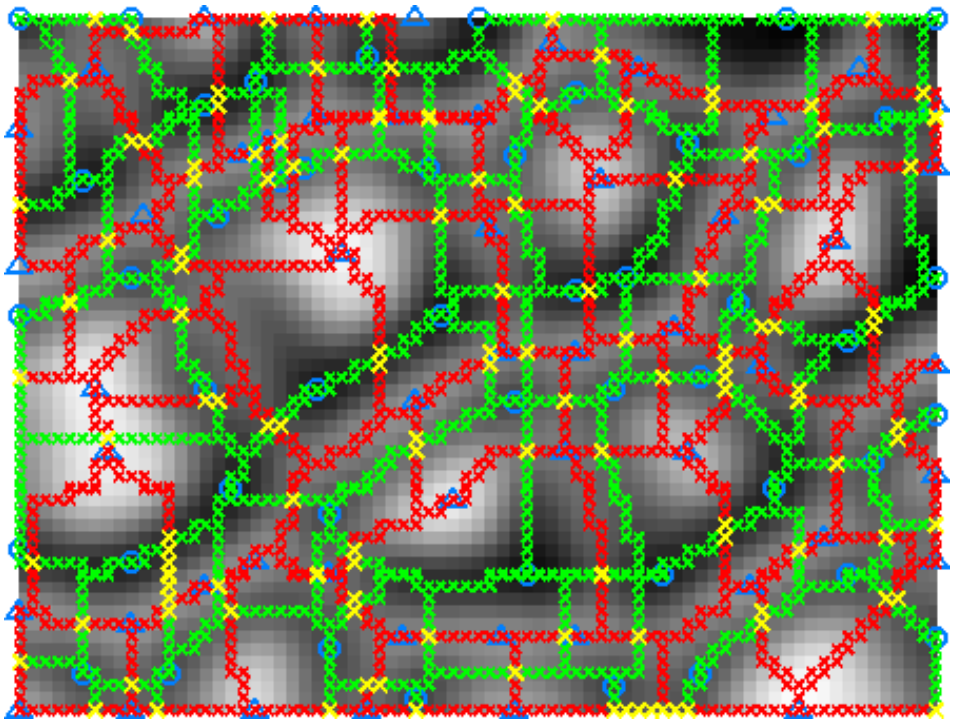
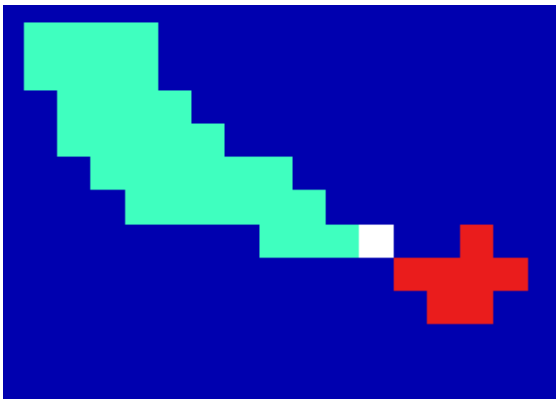
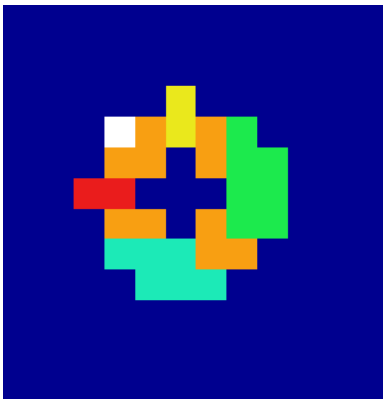


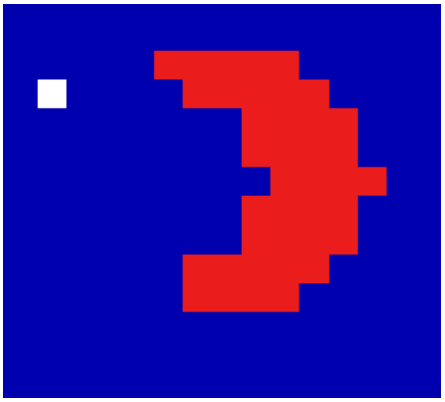
Figure 8: In this example, the blue triangles are the maxima, the blue circles are the minima, the red x's are the watershed, the green x's are the watercourse and the yellow x's are the intersection of the watershed and watercourse pixels. Observe the image boundary pixels after processing the image with mirror reflection padding. Most of the boundary pixels are either critical points or watershed/watercourse pixels. The only exception happens when there is a direct transition from a minimum to maximum along the boundary.



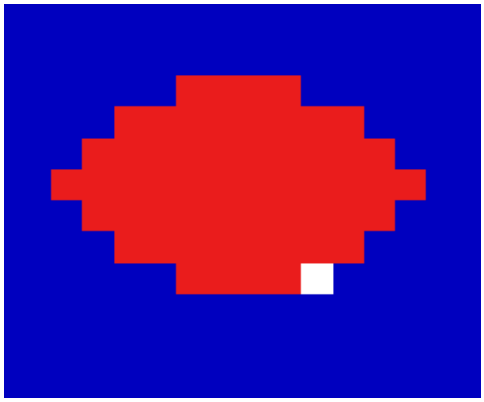
(a) Watershed creation. Since the event pixel is adjacent to two separate basins, it is classified as a watershed pixel.



(b) Watershed growth. The orange pixels are the previously classified watershed pixels and the red, green, cyan and yellow regions correspond to 4 seperate basins. Since the event pixel is only adjacent to the existing watershed pixels, it is classified as a watershed pixel as well.



(c) Basin creation. Since the event pixel is not adjacent to any watershed or basin pixel, it is classified as a new basin (minimum).



(d) Basin growth. Since the event pixel is adjacent to only one basin, it is just a part of that basin. Thus, it is classified as a basin pixel.

Figure 9: Event pixel classification examples. The white pixels are the event pixels to be classified.

iv. Copy non-zero labels from the previous iteration ($Label\_prev$) into $Label\_curr$. So, now $Label\_curr$ has the same labels as $Label\_prev$, but additionally it also has pixels labeled as -1 which correspond to new pixels covered with water, the so-called "event pixels" which will correspond to either minima creation, watershed creation or basin growth.

v. Classify each connected component of event pixels as "min", "watershed" or "basin growth".

(a) [**watershed creation**] Figure 9a. It is a "watershed" if it touches at least 2 separate regions with non-zero labels. Add each pixel of the component to ws. Add the ids of the basins participating in the collision to ws_basins. In this way, we always know which minima give rise to which watershed pixels.

A. [**watershed growth**] Figure 9b. Added this condition recently, not sure about its correctness. It is a "watershed" if it is surrounded by 0s in $Label\_curr$, but at least one of the surrounding pixels has been labeled as "watershed". This case does not happen frequently, but it can still happen. However, I am not sure if this is the best way to handle it. In order to handle this case more properly, the watershed pixels can be represented as -2 in $Label\_curr$ and $Label\_prev$.

(b) [**basin creation**] Figure 9c. It is a "min" if it is surrounded by 0s in $Label\_curr$ and none of the surrounding pixels has been labeled as "watershed". Either fit a surface to find the subpixel position or take the centroid of the component and add it to $mins$. $num\_min = num\_min + 1$.

(c) [**basin growth**] Figure 9d. It is a "basin growth" if it touches only one region with a non-zero label.

vi. Update -1s in $Label\_curr$ with appropriate labels (use label -2 for watershed creation, label num_min for basin creation and the label of the growing basin for basin growth)

vii. $Label\_prev = Label\_curr$.

viii. $i = i + 1$.

ix. Exit the loop if all the water-levels have been visited.

5) Return $mins$, ws and ws_basins.

function $TAG\_detector(I, \sigma)$**:**

Summary: The graph connecting the critical points can only connect saddles to other critical points, *i.e.*, the only connection of maximum and minimum is to saddle point and not to other maximum or minimum. So, we first compute the minima/maxima and watershed/watercourse pixels. Then, based on our definition of saddles, we intersect

the watershed and watercourse pixels and find the saddle candidates. Next, we classify each candidate either as saddle or as a series of saddle points connected as a chain. Finally, we connect all the saddle points to minima, maxima and other saddles based on the identities of minima/maxima giving rise to each watershed pixel. This generates the TAG of I.

Failure mode: I do not have a generalized rule to find saddle-saddle connections. So, the saddle-saddle connections are ignored which causes the algorithm to create incorrect links in the TAG. Show examples.

Input:

- Image I
- σ for smoothing

Output:

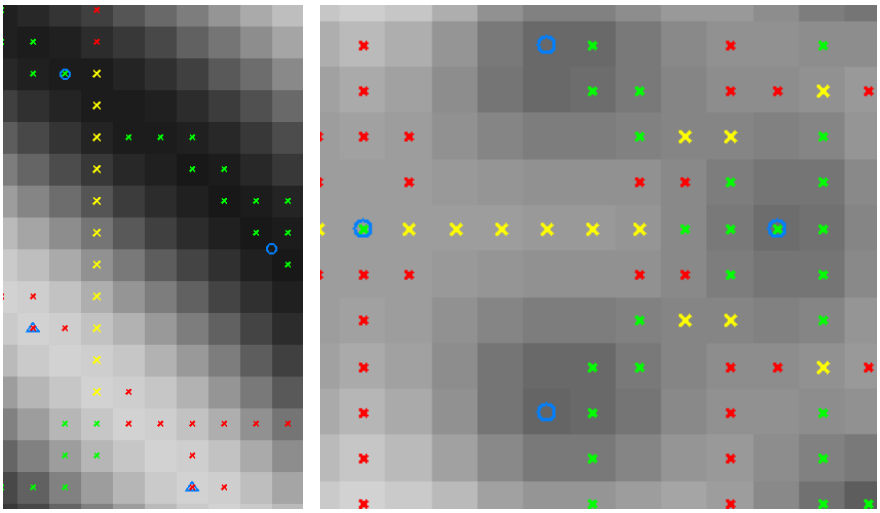
- $mins$: the list of minima of I
- $maxs$: the list of maxima of I
- $saddles$: the list of saddle points of I
- $saddle\_links$: the ids of the minima/maxima/saddle points connected to each saddle. It corresponds to the TAG of I.

Local variables:

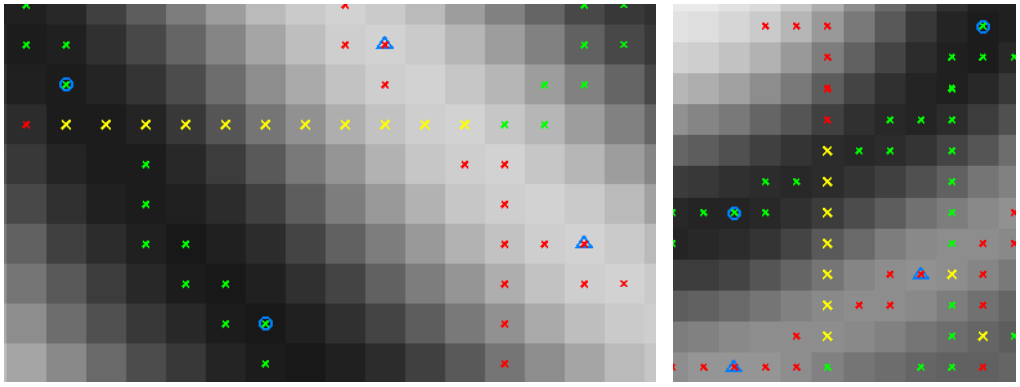
- ws: the unorganized list of watershed pixels of I
- wc: the unorganized list of watercourse pixels of I
- ws_basins: ids of the minima giving rise to each watershed pixel
- wc_hills: ids of the maxima giving rise to each watercourse pixel
- ws_wc_common: common pixels of ws and wc
- $saddle\_candidates$: the list of connected components in ws_wc_common

Algorithm:

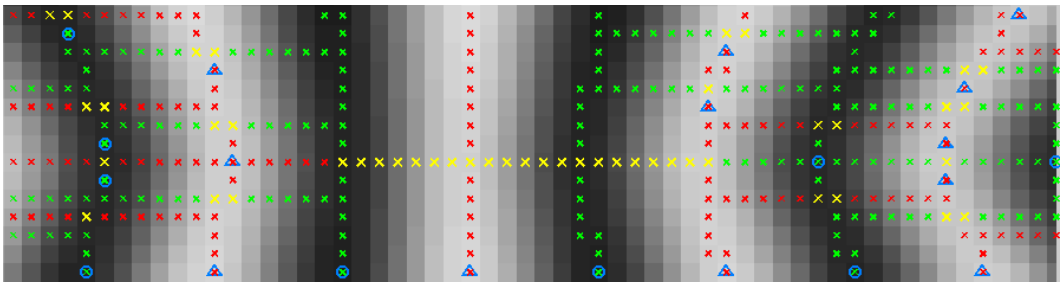
1. Call $min\_watershed\_detector(I, \sigma)$ to compute ws, $mins$ and ws_basins
2. Call $min\_watershed\_detector(-I, \sigma)$ to compute wc, $maxs$ and wc_hills
3. Find the intersection of ws and wc and store these pixels in ws_wc_common. These are saddle points either as isolated saddle points or as chains of saddle points (See Figure 10e). Keep track of the ids in ws_basins and wc_hills and associate them with each pixel in ws_wc_common.



(a) (b)



(c) (d)



(e)

Figure 10: In these examples, the blue triangles are the maxima, the blue circles are the minima, the red x's are the watershed, the green x's are the watercourse and the yellow x's are the intersection of the watershed and watercourse pixels. The yellow chains are the examples of the cases where I think the saddle-saddle connections occur. Note that there are other yellow groups with 1 or 2 pixels, too. Those are the examples of isolated saddle points. Do not confuse them with the saddle chains.

4. Find the connected components in *ws_wc_common* and store them in *saddle_candidates*.

5. Classify and store each connected component in *saddle_candidates* as follows:

   i. If all the pixels of the component have the same min and max ids according to their entries in *ws_basins* and *wc_hills*, fit a quadratic surface to the entire component, find its subpixel critical point and add it to *saddles*. [5]

   ii. Otherwise, this may imply that this component corresponds to at least two saddle points connected by saddle-saddle links. [6] See Figure 10 for examples of this case.

6. Compute *saddle_links* (ids of minima/maxima/saddle points connected to each saddle) based on the component classification results. Basically, use the entries in *ws_basins* and *wc_hills* if it is classified according to Step 5i. Use the connected saddle information if it is classified according to Step 5ii.

7. Return *mins*, *maxs*, *saddles* and *saddle_links*.

6 Use of Critical Points as Features

In this section, we propose the use of critical points as features as an alternative to other features such as the Harris corners [10], SIFT keypoints [17], *etc.* A clear difference between critical points and other features is that the latter are based on local detections, *e.g.*, by optimizing a funcitonal on a local grid. The detection of critical points, especially saddle points, however, is global, and thus more robust. We illustrate two applications using critical points; one in image matching and one in multiview image calibration. Since both applications require matching of these features using some local descriptor, we describe that first.

6.1 Matching critical points

We use the standard methodology for matching features based on a traditional local descriptor as described in [17]. First, the local descriptor is the SIFT histogram. It should be obvious that this descriptor is not optimal for describing critical points as it is tailored to describe SIFT keypoints. We keep the descriptor identical to test the viability of using critical points compared to SIFT and Harris corners. Alternatively, the connectivity of a critical point with its neighbors itself is a more intrinsic and natural descriptor of each critical point. We do not use this descriptor in this paper.

Second, the matching strategy is identical to the one described in [17]. A feature from one of the images is compared to all of the features in the other image. The closest two features are found and their distance ratio is analyzed. If the ratio is less than a

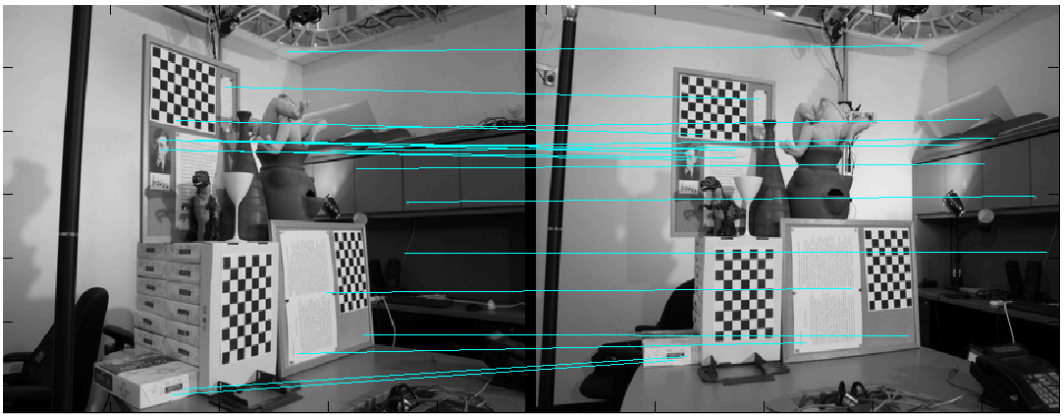
[5] Subpixel computation part is disabled, but there is code to accomplish that. For now, I just average the coordinates of the pixels that belong to the component.

[6] I haven't been able to generalize this case, so it does not work for now.

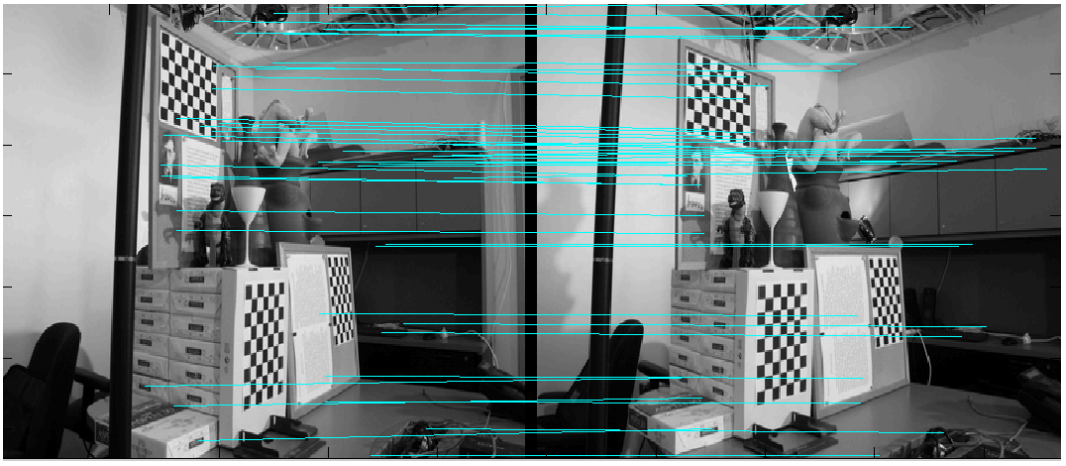
(a) Minima

(b) SIFT keypoints

Figure 11: Comparison of critical points to SIFT points in a narrow baseline setting.

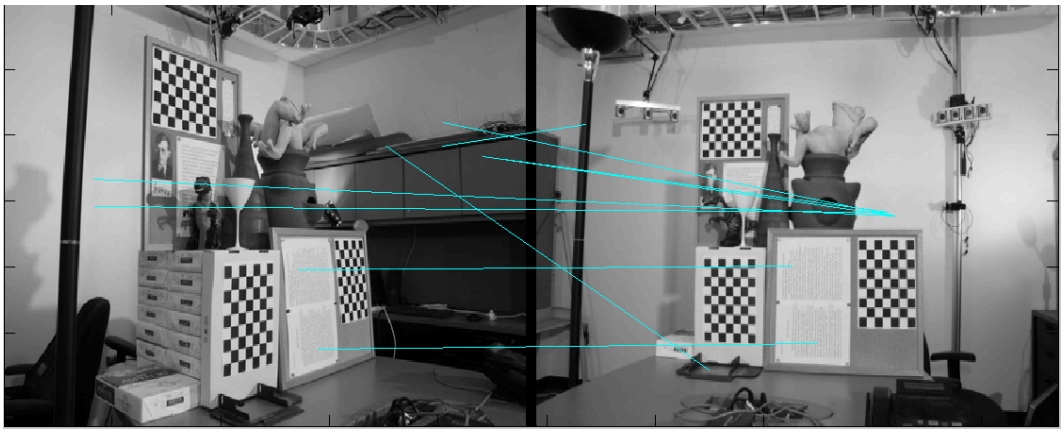


(a) Maxima

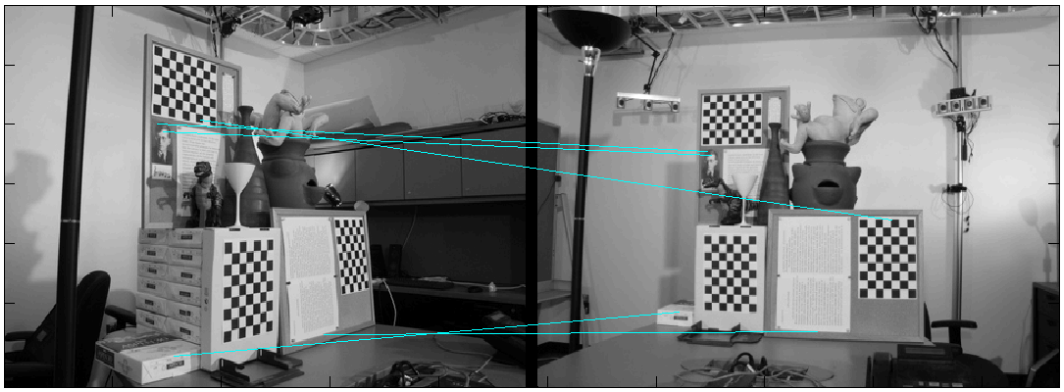


(b) SIFT keypoints

Figure 12: Comparison of critical points to SIFT points in a medium baseline setting.



(a) Saddle points



(b) SIFT points

Figure 13: Comparison of critical points to SIFT points in a wide baseline setting.

certain threshold (usually a value between 0.6 and 0.8), the closest feature is regarded as a correct match.

We use the multiview dataset of [33, 34] which has 13 views of a scene with ground truth correspondence. The critical points in pairs of views are matched as described above and the results are validated using the ground truth correspondence. Figure 11, 12 and 13 show a comparison of using critical points against SIFT keypoints for pairs of images having harrow, medium and wide baselines. Table **??** shows the recall rates for each of these cases. [Evaluation has not been done! I'll try to do it if I have enough time left. the problem is I'm working with resized images since the original ones are too big to be used in my critical point detection algorithm.]

(a) Scene 1

(b) Scene 2

(c) Scene 9

(d) Scene 29

(e) Scene 54

Figure 14: 5 of the 45 scenes we use in our image matching experiments.

6.2 Image matching

Keypoint type	**True positive rate**
Top 100% SIFT	84.89%
Top 100% Min	78.22%
Top 90% Min	77.78%
Top 80% Min	82.67%
Top 70% Min	78.22%
Top 60% Min	78.22%
Top 50% Min	80.89%
Top 40% Min	81.33%
Top 30% Min	80.00%
Top 20% Min	75.11%
Top 10% Min	77.33%
Top 100% Max	75.11%
Top 90% Max	76.00%
Top 80% Max	77.33%
Top 70% Max	78.67%
Top 60% Max	79.11%
Top 50% Max	80.89%
Top 40% Max	79.56%
Top 30% Max	78.67%
Top 20% Max	79.56%
Top 10% Max	75.11%
Top 100% Saddle	77.33%
Top 90% Saddle	79.11%
Top 80% Saddle	78.22%
Top 70% Saddle	79.11%
Top 60% Saddle	79.11%
Top 50% Saddle	80.89%
Top 40% Saddle	82.67%
Top 30% Saddle	83.11%
Top 20% Saddle	83.11%
Top 10% Saddle	80.00%
Top 100% Min + Max + Saddle	78.22%
Top 90% Min + Max + Saddle	78.67%
Top 80% Min + Max + Saddle	80.44%
Top 70% Min + Max + Saddle	80.89%
Top 60% Min + Max + Saddle	80.89%
Top 50% Min + Max + Saddle	83.11%
Top 40% Min + Max + Saddle	82.67%
Top 30% Min + Max + Saddle	84.89%
Top 20% Min + Max + Saddle	83.56%
Top 10% Min + Max + Saddle	84.00%
Top 30% Min + 80% Max + 10% Saddle	87.11%
Top 80% Min + 20% Max + 90% Saddle + 100% SIFT	**88.89**%
Top 10% Min + 10% Max + 20% Saddle + 100% SIFT	**88.89**%

Table 1: Image matching results.

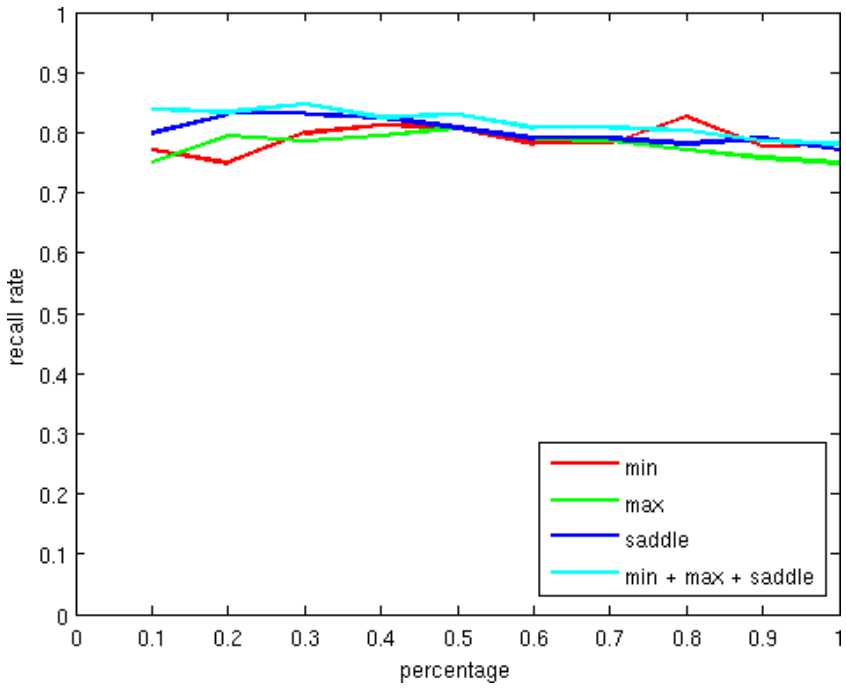


Figure 15: Recall rates for top $T\%$ critical points.

In this section, we describe an application of feature matching using critical points to the problem of image matching. We subsampled the Quarter Size Reduced Robot Dataset [1] which contains 60 scenes and 266 images per scene to create a smaller dataset. This dataset contains 45 scenes and 5 images per scene for a total of 225 images, Figure 14. We used a leave-one-out strategy where a match is correct if the two matching images belong to the same scene. Observe that small amounts of noise can generate numerous spurious critical points. We used the minimum principal curvature[7] of the intensity profile as a measure to indicate the likelihood that a critical point comes from noise, since noise would generate low curvature profiles. With this measure, it is possible to use only the top $T\%$ of critical points ranked order by this measure in the matching task. Table 1 shows the results for various values of T using minima, maxima, saddles and their combinations. These results show that the recall is not so sensitive with changes in T.

A key question is whether a random sampling or a regular grid sampling is more effective than keypoint sampling. Given a fixed number of samples N, we can fairly compare the recall rate for regular samples, random samples, and the top N critical points, Figure 16. Observe that *(i)* for both regular samples and random samples the recall rate increases and saturates with regular sampling doing a little better. *(ii)* The recall rate decreases as the number critical points increases, possibly due to the effect

[7]We tried other curvature measures such as Gaussian and mean, but the minimum curvature yielded the best results.

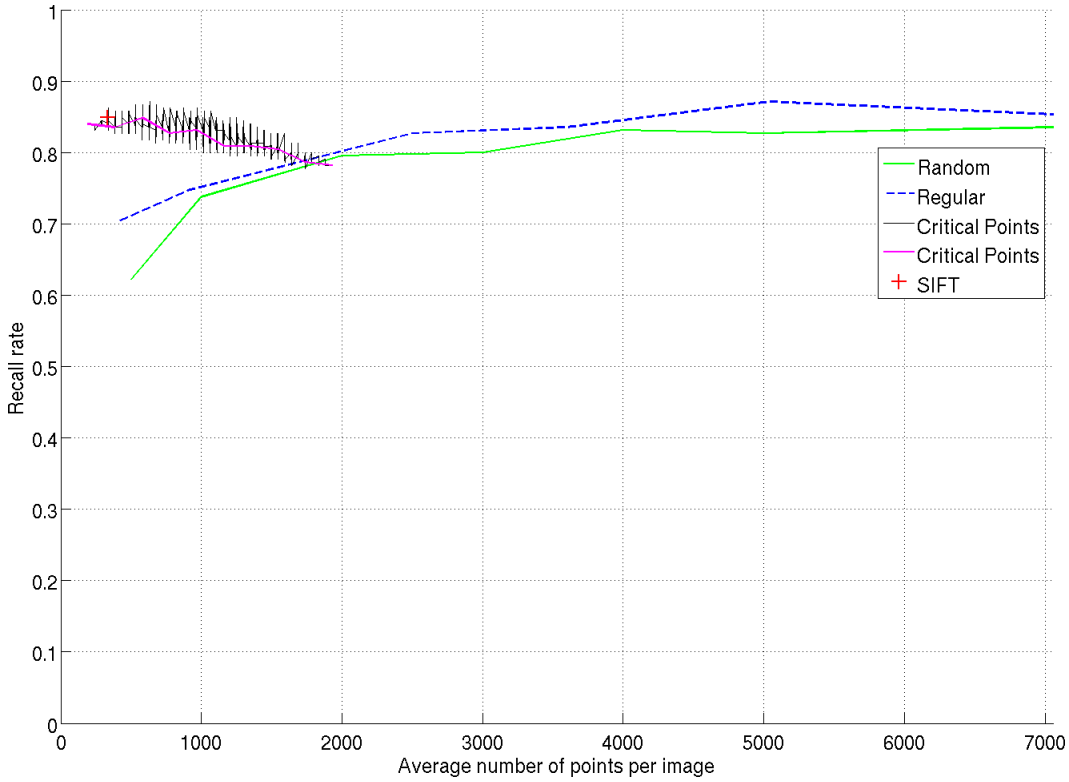


Figure 16: True positive rate vs number of keypoints

of noise. *(iii)* The recall rate for SIFT keypoints is also shown which is roughly in the same ball park with using fewer critical points. This figure concludes that *(i)* feature points are not useful if many random or regular samples can be used and *(ii)* the benefit of feature points is effectively on working with fewer samples.

In conclusion, critical points with a SIFT descriptor achive similar recall rate as SIFT keypoints with a SIFT descriptor. We conjecture that a more natural descriptor for critical points is using a SIFT descriptor but using joint matching of neighbors would lead to substantially better results. For example, if we use the K nearest neighbor matches of two neighboring critical points in image one, we would require that the matches also be neighbors in image 2, Figure **??**.

6.3 Use of critical points in multiview calibration

	SIFT	**Minimum**	**Maximum**	**Saddle**
Scene 1	0.62	0.71	*0.47*	0.48
Scene 2	0.58	0.32	0.28	*0.26*
Scene 3	*0.36*	0.44	0.42	0.44
Scene 4	0.28	*0.11*	0.34	0.31

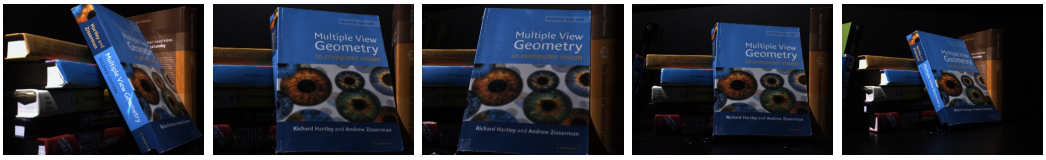
Table 2: Bundler calibration, average reprojection errors in pixels

The traditional approach to calibrating two views is to use RANSAC on potential matches to generate the candidate calibrations and then verify using the remaining points. A useful package for this purpose is Bundler [31] which takes an input of features and uses Lowe's algorithm to generate matches [not sure, have to check!!!]. Since Bundler expects SIFT keypoints we had to customize the code to allow use of critical points instead. The reprojection errors measure the effectiveness of two competing calibrations. We used a dataset of four scenes and 14 multiview images of each with a constant illumination setting [future todo: study the effects of varying illumination], Figure 17.

Table 2 compares the average reprojection error when using SIFT with using one type of critical point, *i.e.*, min, max, and saddle, respectively. We have not shown the result of using all critical points because Bundler does not allow a representation to type, *e.g.*, to prevent matching a minimum to a maximum.

The results show that in all but Scene 3, one type of critical point performs better than SIFT; and, in the case of Scene 3 the results are not sognificantly worse. We expect better results when all types are combined. As it is, using a set of four bundle adjustments one for each feature type, and using the one with minimum reprojection error leads to significant improvement.

(a) Scene 1

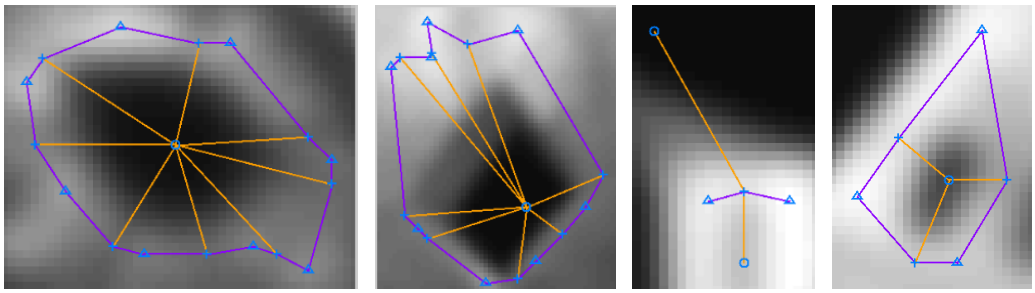


(b) Scene 2

(c) Scene 3

(d) Scene 4

Figure 17: The scenes used in the calibration experiments.



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

Figure 18: Local TAG structures. These graphs can be used to describe the critical points.

7 Future Directions

7.1 Local TAG structures as feature descriptors

As shown in Section 6, critical points are promising interest points. In our experiments, we treated each critical point as a standalone interest point represented by a SIFT descriptor. However, there exists a more natural way to describe critical points: TAG.

TAG captures the local neighborhood structure of critical points which seems to be a distinctive representation. See Figure 18 for examples of local TAG structures. Our idea for computing the dissimilarity of two critical points is as follows: Given two local TAG structures, we plan to apply certain graph edit operations to convert one of the graphs to the other. The resulting edit distance can then be used as a matching score or dissimilarity. Some possible graph edits are

- Global transformations (rotation and scaling)
- Edge deletion
- Individual edge rotation
- Individual edge scaling
- Intensity change
- Curvature change.

7.2 Level-set implementation

The current algorithm is processing the water levels using a discrete process which has its own problems. For example (see Figure 10e), in our discrete process, the watershed (or watercourse) lines sometimes form junctions without any critical points. It does not make any sense, since there must pass only one slope line from a regular point.

Prof. Kimia believes that this kind of artifacts due to discreteness might be avoided in a level-set framework.

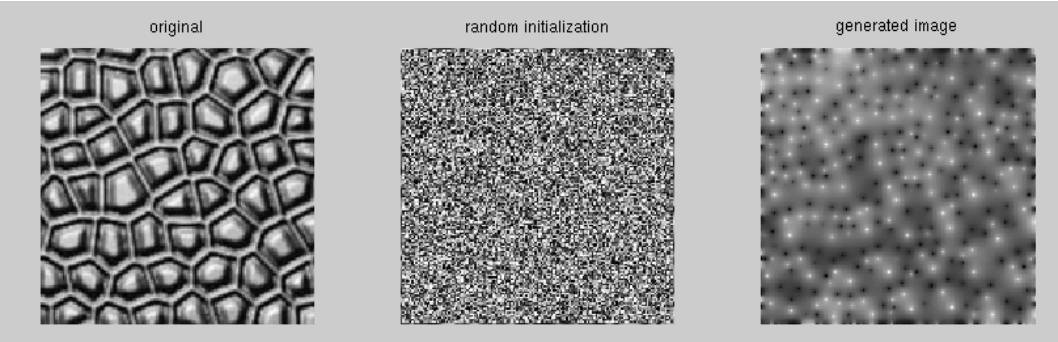
The proposed idea is to visit the water levels by evolving a level-set function according to the underlying intensity values. Prof. Kimia believes that their paper on "shape form shading" [14] has a similar level-set formulation.

7.3 Organizing the watershed/watercourse pixels

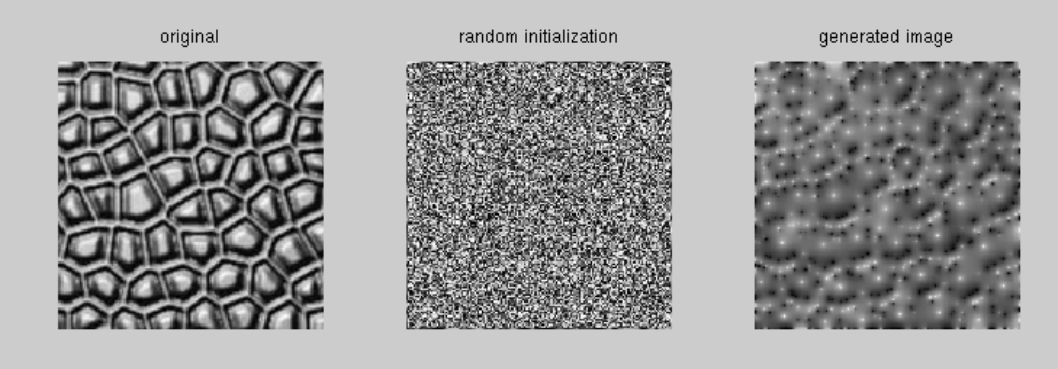
The current algorithm produces a set of unorganized watershed and watercourse pixels. However, organizing them as curves might help understand some of the problems we are facing (*e.g.*, the watershed junctions making no sense). In addition, it will be useful in getting the subpixel watershed lines.

7.4 Image generation

We had explored the possibility of generating images from given TAGs. Our method for image generation was fixing the intensity values at certain pixels and running the diffusion equation on the image. See the examples in Figure 19.



(a) Fixed values at minima and maxima.



(b) Fixed values at minima, maxima and saddles.

(c) Fixed values at watershed and watercourse pixels.

Figure 19: Image generation with diffusion.

8 Appendix

Proposition 8 *The curvatures of a slope line and a level set contour are*

$$\kappa_\alpha = -\frac{(f_x^2 - f_y^2) + f_x f_y (f_{yy} - f_{xx})}{(f_x^2 + f_y^2)^{\frac{3}{2}}},$$

$$\kappa_\beta = \frac{f_y^2 f_{xx} - 2 f_x f_y f_{xy} + f_x^2 f_{yy}}{(f_x^2 + f_y^2)^{\frac{3}{2}}},$$

respectively.

Let $f(x, y) = c_0$, where c_0 is a constant be a level set curve of β of f. Then, by differentiating this with respect to arclength $\tilde{s}$ along β, we have

$$\nabla f . T_\beta = 0,$$

so that $T_\beta = \frac{\nabla f^\perp}{|\nabla f|}$. Differentiating again we have

$$\frac{(\nabla f^\perp)^T}{|\nabla f|} H T_\beta + \nabla f (\kappa_\beta N_\beta) = 0,$$

from which κ can be obtained using $N_\beta = \frac{-\nabla f}{|\nabla f|}$ as

$$\kappa_\beta = \frac{\frac{1}{|\nabla f|^2} (\nabla f^\perp)^T H \nabla f^\perp}{\nabla f \frac{\nabla f}{|\nabla f|}} = \frac{f_y^2 f_{xx} - 2 f_x f_y f_{xy} + f_x^2 f_{yy}}{(f_x^2 + f_y^2)^{\frac{3}{2}}}. \tag{2}$$

The contour line curvature can also be expressed in (u, v) coordinates as

$$\kappa_\beta = \frac{f_{vv}}{f_u}. \tag{3}$$

Similarly, for a slope line, the tangent $T_\alpha = \frac{\nabla f}{|\nabla f|}$ and $N_\alpha = \frac{\nabla f^T}{|\nabla f|}$ we have

$$\nabla f^\perp . T_\alpha = 0.$$

Differentiating this with respect to arclength along α we have

$$\frac{(\nabla f^\perp)^T H}{|\nabla f|} T_\alpha + \nabla f^\perp (\kappa N_\alpha) = 0$$

from which κ can be determined as

$$\kappa_\alpha = -\frac{\frac{(\nabla f^\perp)^T H \nabla f}{|\nabla f|^2}}{\nabla f^\perp \frac{\nabla f^\perp}{|\nabla f|}} = -\frac{(f_x^2 - f_y^2) + f_x f_y (f_{yy} - f_{xx})}{(f_x^2 + f_y^2)^{\frac{3}{2}}}. \tag{4}$$

This can also be written as

$$\kappa_\alpha = -\frac{f_{uv}}{f_u}. \tag{5}$$

References

[1] Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. Interesting interest points - a comparative study of interest point performance on a unique data set. *International Journal of Computer Vision*, 97(1):18–35, 2012.

[2] Peer-Timo Bremer, Herbert Edelsbrunner, Bernd Hamann, and Valerio Pascucci. A topological hierarchy for functions on triangulated surfaces. *IEEE Trans. Vis. Comput. Graph.*, 10(4):385–396, 2004.

[3] A. Cayley. On contour and slope lines. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, 18:264–268, October 1859.

[4] Supoj Chinveeraphan, Ryo Takamatsu, and Makoto Sato. A hierarchical description of digital grayscale images based on image dipoles. In *International Conference on Pattern Recognition*, pages 246–250, Vienna, Austria, 1996. Computer Society Press.

[5] James Damon. Generic properties of solutions to partial differential equations. *Arch. Rat. Mech. Anal.*, 1995.

[6] James Damon. Generic structure of two-dimensional images under gaussian blurring. *SIAM Journal on Applied Mathematics*, 59(1):97–138, 1998.

[7] Paul Emile Breton de Champ. Note sur les lignes de faîte et de Thalweg. *C.R. Acad. Sc. Paris*, 39:647, 1854.

[8] Guoyi Fu, S. A. Hojjat, and Alan C. F. Colchester. Integrating watersheds and critical point analysis for object detection in discrete 2D images. *Medical Image Analysis*, 8(3):177–185, 2004.

[9] Steve Gu, Ying Zheng, and Carlo Tomasi. Critical nets and beta-stable features for image matching. In *Proceedings of European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 663–676. Springer, 2010.

[10] Chris Harris and Mike Stephens. A combined edge and corner detector. In *Alvey Vision Conference*, pages 189–192, 1988.

[11] Camille Jordan. Sur les lignes de faîte et de thalweg. *C.R. Acad. Sc. Paris*, 74(1457):705–710, 1872.

[12] Fırat Kalaycılar and Benjamin Kimia. A topographical and topological representation of images. `http://vision.lems.brown.edu/techreports/ttri`.

[13] Yannick L. Kergosien and Rene Thom. Sur les points paraboliques des surfaces. *C.R. Acad. Sc. Paris*, 290(15), 1980.

[14] Ronnie Kimmel, Kaleem Siddiqi, Benjamin B. Kimia, and Alfred M. Bruckstein. Shape from shading: Level set propagation and viscosity solutions. *International Journal of Computer Vision*, 16(2), October 1995.

[15] Jan J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[16] Yanxi Liu, Wen-Chieh Lin, and James Hays. Near-regular texture analysis and manipulation. *ACM Trans. Graph.*, 23(3):368–376, 2004.

[17] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[18] James C. Maxwell. On hills and dales. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science XL*, pages 421–427, 1870.

[19] Lee R. Nackman. Two-dimensional critical point configuration graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(4):442–450, 1984.

[20] John L. Pfaltz. Surface networks. *Geographical Analysis*, 8(1):77–93, 1976.

[21] Sanjay Rana. Novel structural analyses of surface networks. In *Proceedings of GeoComputation Conference*, pages 1–7, NUI Maynooth, Ireland, 2007.

[22] Joachim H. Rieger. Generic properties of edges and corners on smooth greyvalue functions. *Biol. Cybernet*, 66:497–502, 1992.

[23] Joachim H. Rieger. Generic evolutions of edges on families of diffused greyvalue surfaces. *J.Math.Imaging Vision*, 5:207–217, 1995.

[24] Joachim H. Rieger. Topographical properties of generic images. *International Journal of Computer Vision*, 23(1):79–92, 1997.

[25] Paul L. Rosin. Early image representation by slope districts. *Journal of Visual Communication and Image Representation*, 6(3):228–243, 1995.

[26] Paul L. Rosin, Alan C. F. Colchester, and David J. Hawkes. Early image representation using regions defined by maximum gradient paths between singular points. *Pattern Recognition*, 25(7):695–711, 1992.

[27] M. De Saint-Venant. Surfaces á plus grande pente constituées sur des lignes courbes. *Bulletin de la soc. philomath. de paris*, pages 24–30, 1852.

[28] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE Computer Society, 2007.

[29] Shaadi Shidfar and Alan Colchester. Detection and correction of invalid slope districts in separatrix-based image segmentation. In *Machine Vision and Image Processing (MVIP), 2011 7th Iranian*, pages 1–6. IEEE, 2011.

[30] Shaadi Shidfar, Adina Ion, Lazaros Ktorides, and Alan Colchester. Improvements in a generic approach to low level data-driven segmentation based on separatrices. In *Proceedings of the BMVC 2010 UK postgraduate workshop*. BMVA Press, 2010.

[31] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006.

[32] Shigeo Takahashi, Tetsuya Ikeda, Yoshihisa Shinagawa, Tosiyasu L. Kunii, and Minoru Ueda. Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data. In *Computer Graphics Forum*, volume 14, pages 181–192. Wiley Online Library, 1995.

[33] Anil Usumezbas and Benjamin B. Kimia. Multi-view dense point correspondence ground-truth dataset. `http://vision.lems.brown.edu/datasets/dense-corr`.

[34] Anil Usumezbas and Benjamin B. Kimia. Generating dense point correspondence ground-truth across multiple views. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, 3DIMPVT 2012*, pages 1–8, Zurich, Switzerland, October 2012.

[35] Tino Weinkauf and David Guenther. Separatrix persistence: Extraction of salient edges on surfaces using topological methods. *Computer Graphics Forum (Proc. SGP '09)*, 28(5):1519–1528, July 2009.

[36] Wikipedia. Picard-Lindelöf Theorem. `http://en.wikipedia.org/wiki/Picard%E2%80%93Lindel%C3%B6f_theorem`.

[37] Wikipedia. Watershed (image processing). `http://en.wikipedia.org/wiki/Watershed_(image_processing)`.

[38] Joseph Wood. Constructing weighted surface networks for the representation and analysis of surface topology. In *5th International Conference on GeoComputation*, pages 23–25, Chatham, UK, September 2000.