ABSTRACT

Title of dissertation: THE IMAGE TORQUE OPERATOR FOR MID-LEVEL VISION: THEORY AND EXPERIMENT

Morimichi Nishigaki, Doctor of Philosophy, 2012

Dissertation directed by: Professor Yiannis Aloimonos
Department of Computer Science

A problem central to visual scene understanding and computer vision is to extract semantically meaningful parts of images. A visual scene consists of objects, and the objects and parts of objects are delineated from their surrounding by closed contours. In this thesis a new bottom-up visual operator, called the *Torque* operator, which captures the concept of closed contours is introduced. Its computation is inspired by the mechanical definition of torque or moment of force, and applied to image edges. It takes as input edges and computes over regions of different size a measure of how well the edges are aligned to form a closed, convex contour. The torque operator is by definition scale independent, and can be seen as an operator of mid-level vision that captures the organizational concept of 'closure' and grouping mechanism of edges. In this thesis, fundamental properties of the torque measure are studied, and experiments are performed to demonstrate and verify that it can be made a useful tool for a variety of applications, including visual attention, segmentation, and boundary edge detection.

THE IMAGE TORQUE OPERATOR FOR MID-LEVEL VISION: THEORY AND EXPERIMENT

by

Morimichi Nishigaki

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2012

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Yiannis Aloimonos, for giving me an invaluable opportunity to study on computer vision under his advices. I have learned from him many interesting theories and techniques related to my research in this opportunity. He has given me many suggestions and advices on incredibly wide range of topics.

I would also like to express my deep appreciation and gratitude to my co-advisor, Dr. Cornelia Fermüller. Without her extraordinary accurate theoretical advices, this thesis would not be accomplished. She has always shown me the direction of my research and her instructions has been the guidance of my research.

My colleagues at the computer vision laboratory have enriched my graduate life in many ways. Discussions with them have been valuable to me, and often new knowledge has come from those discussions.

I would like to thank my family. My mother and father have always stood by me. They devoted themselves to my education, and that has been the foundation of my further study toward Ph.D. My relatives have also supported me in variety ways and encouraged me in the challenging situations.

I owe my deepest thanks to my wife, Wei-Lun. She has supported me for my study and research over past many years since we married. She devoted herself to me for maintaining the most comfortable environment to pursue Ph.D.

I apologize to those I have inadvertently left out. Lastly, I would like thank all the people who have made this thesis possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Visual Cues

The problem of understanding the visual scene has been central to computer vision. The classical approach to scene understanding [1] has been to first apply processes that segment the scene into different surfaces and make models of it, the so-called 'reconstruction processes', which then apply processes of recognizing the different parts of scene. Researches on reconstruction of visual scene aim to estimate the physical parameters of the visual scene, such as depth, motion (scene motion and/or ego-motion), the boundary of objects, the direction of the light source, the surface reflectance, and so on. In order to reconstruct visual scenes, the problem of inverse optics has to be solved. However, since the inverse optics problem is ill posed, specific assumptions have to be made to reduce the ambiguities. For example, knowledge of the reflectance map in shape from shading, rigidity in structure from motion, or a 3D texel distribution in shape from texture has been assumed.

Recently, segmentation-based approaches have attracted attention for tackling the problem. Especially color segmentation-based methods for the estimation of optical flow and stereo disparity have gained in popularity, because of their good performance. These segmentation approaches divide the image on the basis of color-coherency into usually a large number of regions. The main issue for optical flow and stereo disparity estimation is the image correspondence problem. It is known that the problem is difficult in some image regions, such as regions of poor texture and at the boundaries of objects. Color segmentation-based approaches have advantages in these regions, because the color segmentation provides information on surfaces and their boundaries. The underlying assumptions are: First, similarly colored neighboring pixels have similar depth values or motions. In other words, the depth or motion of all image points within a segment is well approximated by a model, such as the affine or planar model. Second, the discontinuities in depth or motion coincide with the boundaries of the segments, so that color discontinuities can be used to delineate depth boundaries or motion discontinuities. Most of the segmentation-based approaches start with an over-segmentation to ensure these assumptions. However, the assumptions are not always true.

A wide range of computer vision problems could in principle make good use of segmented images, as has been demonstrated with recent segmentation-based approaches. This is presumably related to the fact that perceptual grouping plays a powerful role in human visual perception. However, the problem of image segmentation and grouping itself remains a great challenge for computer vision. For once, there does not exist a clear definition of image segmentation, because segmentation is related to semantics. Thus in many situations there is not a single correct answer. Several interpretations may be possible depending on the context, and the partitioning is inherently hierarchical.

Early vision is considered a collection of modules which deal with the estimation of quantities in the coordinate system of the observer, such as stereo disparity (depth), optical flow (image motion), texture, occlusion, contour, and various segmentations (color, depth, motion). These visual cues are fundamental primitives for vision processing. It has long been known that these modules are intertwined in a 'Chicken and Egg' fashion [2]. The issue is, what kinds of visual cues are useful and how these visual cues are integrated. With an awareness of these two issues, we focus on the first issue in this thesis.

Visual cues are categorized into low-, mid- and high-level cues. This categorization is a rough distinction based on the level of abstraction, ranging from the raw image to semantic recognition. Mid-level visual cues includes contour grouping, region grouping, and figure/ground organization. Evidence from psychophysics suggests that vision processes on mid-level visual cues are important components of the visual system and heavily interact with object and scene recognition.

In this thesis we introduce a scale-invariant mid-level representation of contour grouping that operates on the raw image. We apply this representation to a number of visual processes, such as boundary detection, visual attention, and segmentation, which are essential for object and scene recognition.

1.2 Mid-level Vision

A central problem for visual scene interpretation is the localization and segmentation of objects in the visual scene. The detection of their boundaries is essential for this purpose. All visual processing starts with the cluttered two-dimensional image(s), which are formed as the projections of the three-dimensional world. Psychologists of the early twentieth century argued that human vision organizes the image clutter at the early stages of interpretation through a process of figure ground segmentation by identifying the object-related image regions for further processing. They suggest that certain principles are applied to group pieces of an image and locate borders of figures. Most theorists of vision argue that scene interpretation involves processes at different levels of abstraction, which are categorized into low, mid and high level vision. Low level processes compute features, such as local edges, color, texture, image motion, and stereo disparity. Mid-level processes group the features into meaningful geometric structure in the image, using cues such such as convexity, closeness and parallelism of edges. They combine the low-level features into larger coherent patches to obtain surfaces and other information such as 3D structure, 3D motion, and lighting. Finally high level vision utilizes semantic information to recognize objects, actions and scenes. Many will agree that mid-level vision to a large degree is about implementing organizational principles, some of which have been proposed by the Gestalt theorists.

Not many studies, however, have explored mid-level cues as a tool for boundary detection and segmentation. Some mid-level cues explored on contours are junctions [3], parallelism, line continuity [4], and convexity [5, 6, 7]. Recently the focus has shifted to data-driven approaches [8] to acquire mid-level representations; for example [9] learns mid-level cues from low level cues.

1.3 Boundary

Contours are an important cue for segmentation and recognition. By the term contour, we generally refer to extended curve or edge fragments which present some meaningful geometric concept. As a tool for recognition, so called contour patches, which are local descriptors of spatial edge distributions have been developed. For example, the shape context descriptor [10] encodes the spatial distribution of edge points in log polar space, or [11] defines a feature detector based on the saliency of local convexity. Recently, data driven approaches have become more popular, which acquire contour fragments and their detectors [12, 13, 14] using sophisticated learning techniques from large amounts of data.

Contour detection often is related to figure-ground segmentation. Local cues, such as brightness, color, texture, and their gradients are combined, and weights for each cue are learned using training image data sets with ground truth contours [15, 16]. Recent high performance contour detection algorithms have an additional globalization process combining local cues based on the affinity of distant pixels [17].

Leichter *et al*. [18] proposed a CRF based method to determine boundary ownership for a set of given curves marked along object boundaries. Its main novelty is the use of a 2.1D model, that represents the depth ordering of image segments, and the boundary owner ship is deduced from the depth ordering. The CRF model consist of curve-related potential functions, junction-related potential functions, and potential functions related to non-adjacent segment pairs. The curve-related potential functions are defined as the negative log posterior of the two segments' ordinal depth values conditioned on the curve cues. The posterior is proportional to the cue likelihood. Cues considered here are convexity, lower region, fold/cut and parallelism. The junction-related potential functions are defined similarly, where cues are angles corresponding to segments at the junctions. A set of segment pairs where the potential function's related non-adjacent segments are defined, include segment pairs which have similar hue distribution near curves. Their experimental results show that their method outperform the previously best method.

1.4 Visual Attention

Visual attention or saliency can be used as pre-process for foreground segmentation. Attention mechanisms are classified into bottom-up and top-down processes. Top-down attention is more complex because it represents objects in memory [19] and uses the memory to detect likely objects in an attended visual scene. Bottom up attention is driven by low level processes. Probably the best known model of visual attention is the one proposed by Itti *et al.* [20]. In this model, first local feature maps are computed from color, intensity and orientation information as the difference of Gaussian filters at multiple scales, which approximate the 'center-surround' differences of neurons. Larger center surround differences are considered more 'conspicuous'. Then a combined saliency map is constructed by adding the normalized feature maps. Related approaches differ in the choice of feature vectors and combination of features. Harel *et al.* [21] compute a saliency map based on the dissimilarity of features in regions using a graph-based approach. They define a

Markov chain on a fully-connected directed graph with edge weights depending on the dissimilarity of features and distance between the nodes corresponding to the feature points. Then, it is solved for an equilibrium distribution. The same graph-based method is used for normalization as well. The process accumulates mass at nodes that have a high dissimilarity with their surrounding nodes. The authors evaluated the performance of their detector on its ability to predict human attention using the human fixation data of Einhäuser *et al.* [22], and reported that 98% of the ROC area of human based control is achieved, while the model by Itti *et al.* achieved 84%. Recent work in fixation and attention [23, 24] offers an alternative to the traditional "early" feature saliency theories. Based on systematic psychophysical experiments [23] suggests that observers look at objects as if they knew them before they became aware of their identity, and [24] shows that the hypothesis that humans attend to objects has a better predictive power in the data than any other theory.

1.5 Segmentation

Segmentation involves combining information from the interior of surfaces, that is regions which are smooth in some quantities, such as color intensity and texture [25, 26], motion [27, 28] or depth [29, 30] with information from the discontinuities in these quantities, which are reflected as luminance edges and discontinuities in the texture and the geometric cues. Most approaches treat the problem of segmentation as dividing the image into multiple regions using different clustering techniques.

Examples are the mean-shift method [31] that first smoothes the image using non-parametric parameter estimation and then clusters, and graph-partitioning methods using the graph cut algorithm [32] or the normalized cut [26].

The methods closer related to segmenting objects are those that consider the problem of segmentation as separating one foreground object from background. The problem is usually modeled as optimizing for a binary labeling that assigns each pixel a label, such that the labeling is both consistent with the observed data and piecewise smooth. One sets up an energy model, which either is formulated probabilistically and solved using belief propagation or graph cut methods [33, 29], or continuous and reformulated with differential equations and solved using active contours and variational approaches [34, 35, 36]. An interesting biological motivated formulation was introduced in [37], which segments in the polar coordinate system by minimizing for a closed contour surrounding a fixation point using a graph cut formulation defined on edges only. However, all approaches have the problem of being biased, usually towards small regions with small and smooth boundary. This is because of texture edges, which in real images are always present, in conjunction with minimizations biased towards certain shapes. For example graph cuts [29] are known to favor small areas, the polar coordinate representation favors circular blobs, and variational minimizations [38] explicitly minimize the length and/or smoothness of the bounding contour.

In the context of segmentation, one is interested in the contours that form the boundaries which separate foreground from background. Great advances in contour detection have been achieved through data-driven approaches, championed in the work of Martin *et al*. [15]. In this work, local cues, such as brightness, color, texture, and their gradients are combined, and weights for each cue are learned using image data sets with ground truth contours [15, 16]. In similar spirit [39] learns edge classifiers from simple features in image patches, Ren [40] combines information of local operators from multiple scales, and the high performance contour detection algorithm in [17] includes a globalization process to combine local edges based on the affinity of distant pixels.

A wide variety of figure-ground segmentation methods have been proposed. Prior research can be classified by the dominant visual cue that is used. Most techniques are based on appearance, such as color, intensity and texture [25, 41, 26]. User interactive methods utilize seeds given manually, such as strokes or bounding boxes, to produce a prior for the segmentation. Motion is another useful cue for segmentation in video [27, 28]. Methods using depth information give robust object segmentation [29, 30] at depth boundaries, however depth is not always available.

Segmentation approaches usually are based on local cues (edges and intensity) as input to a global optimization, but recently many methods first compute super-pixels [42, 43] by over-segmenting the image into perceptual uniform regions based on the statistics in neighborhoods or affinity between points.

Taking another viewpoint, figure-ground segmentation methods can also be classified according to whether prior knowledge has been assumed or not.

Gould *et al*. [44] proposed a method to perform both semantic and geometric multi-class image segmentation. An energy function is defined as an horizon potential, an individual region potential and two inter-region potentials. The individual region and the inter-region potentials are defined by multi-class logistic classifiers for semantic and geometric labels. The classifier is trained using extracted features. Features are comprised of raw image features and learned boosted features. The raw image features are the 17-dimensional color and texture features described in [45]. The learned boosted features are scores from boosted classifiers which predict a label given the raw image features. They use a two-stage hill climbing approach to optimize the energy function. In the first stage, pixel-region association variables are modified, then in the second stage, region and horizon variables are optimized. They use max-product BP to infer region classes and geometry variables, then the horizon is updated using ICM (Iterated Conditional Modes). They assume the image was taken by a camera with the horizon axis parallel to the ground.

Schoenemann *et al.* [46] proposed a method both for figure/gound segmentation and inpainting. Usually the boundary length is used as regularity term in the energy function. However, they added curvature regularity to it. Surface continuation constraints and boundary continuation constraints are taken into the optimization, and the problem is formulated into an integer linear program. Since solving integer linear programming is an NP-hard problem, it is solved by linear programming relaxation. Experimental results show the importance of curvature regularization in both segmentation and inpainting.

Hochbaum *et al.* [47] proposed an algorithm to solve the co-segmentation problem by graph-cut that is faster than previous approaches. Co-segmentation is a figure/ground type segmentation, where multiple images of the same or similar objects are given as the foreground object. The co-seg problem is written in a linear combination of MRF minimization and similarity maximization. Prior work [48], however, minimize the difference instead of maximizing similarity. In prior work, the minimization is accomplished by incrementally improving one of the segmentations keeping the other fixed, and the process starts with a given initial segmentation. The proposed algorithm gives a solution without iteration by using the min-cut algorithm on a graph, which has been constructed such that the min-cut method minimizes the cost function.

Lempitsky *et al.* [49] proposed an image segmentation method, that assumes a given bounding box. The authors take tightness into account, where every side of the foreground object is supposed to be close to the bounding box. Tightness constraints are added to the usual energy function consisting of unary and smoothness terms, which turns the minimization to an integer program (IP). The IP, however, is an NP-hard problem, therefore, the problem is relaxed to a linear program (LP) by replacing the integrality constraints by real value constraints. Furthermore, a new scheme to solve the LP problem by iterations is proposed since the LP problem is hard to solve due to the combinatorial number of constraints. The authors pointed out that there is a link between tightness and connectivity. Since the solution of the LP problem is fractional (real value), rounding is necessary to obtain the integer solution. The authors proposed a new rounding algorithm called pinpointing. Given a real-valued priority map, the pinpointing algorithm gives a feasible optimal solution to the problem. The priority may be given by the fractional solution of LP, but it is arbitrary. Experimental results show that the proposed method gives very good results.

1.6 Motion Segmentation

Motion segmentation methods often use optical flow as visual cue. Many methods for optical flow computation have been proposed, but according to the recent benchmarking [50], derivatives of variational methods, originally introduced by Horn and Schunck [51], outperform other approaches. Here we discuss three prominent algorithms for optical flow computation. The optical flow algorithm proposed by Brox *et al.* [52] is categorized as a variational method, where an energy (or cost) function consisting of a data and a smoothness terms is defined, and then the optimal optical flow is found by solving Euler-Lagrange equations. Grey value (or brightness) constancy and gradient constancy assumptions are used in the data term, and the smoothness term is based on the norm of the optical flow gradient. The so-called 'robust norm' function is used instead of quadratic penalisers. They use a multi-scale approach, and the linearization is postponed by computing incrementally the flow. This way, the algorithm is capable of estimating relatively large image motion. The paper is heavily referenced by other optical flow papers. The reason probably is that it gives the details of the numerical computation for solving the Euler-Lagrange equations. Wedel *et al.* [53] proposed another variational optical flow method consisting of a data term and a smoothness term. The data term contains not only the of brightness constraint, but also the epipolar constraint, which is represented by the fundamental matrix. They transform the original variational energy functional into a convex dual form, and then solve the minimization iteratively alternating on the smoothness term and the data term. For the data term minimization, the absolute functions are replaced by dual variables with inequality constraints. They proposed an efficient algorithm to solve the optimization of the data term. Thereby, all possible combinations of the data terms are checked and the optimal solution is found. Their experimental results show that their algorithm was the best at the time of publication. Since they use the epipolar constraint, the performance gets worse when the scene is dynamic. Reference [5] shows how to transform the original energy function to dual form. Another variational optical flow algorithm consisting of data term and smoothness term was proposed by Zimmer *et al*. [54] . They modified the data term by a normalization incorporated into the gradient constancy assumption, and they postpone the linearization using invariant color space and using robust penalisers. The smoothness term is modified by a Joint Image- and Flow-driving regularization (JIF), and JIF is modified by a regularization tensor, rotational invariance, and single robust penalization. The optimal solution is found by solving the Euler-Lagrange equations. They use a coarse-to-fine multi-scale warping approach, where small flow increments are computed via a linearized approach on each warping level. Their experimental results shows that their method is ranked as the best in averaged angular error at the time of publication.

Kampel *et al*. [41] proposed a robust background modeling and shadow suppression, and applied it to motion segmentation. They use Improved Hue, Luminance and Saturation (IHLS). Since hue is an angular value, an appropriate circular statistics should be used. There is a tight relationship between the chrominance component, and hue and saturation. Therefore, given n pairs of observations of hue and saturation, $\left(\theta_i^H, s_i\right)$, the mean chrominance vector is computed based on a saturation-weighted statistics, $C_s = \sum_{i=1}^{n} s_i \cos\theta_i^H$, $S_s = \sum_{i=1}^{n} s_i \sin\theta_i^H$. Given background image data, mean luminance μ_y and associated standard deviation σ_y are computed together with the mean chrominance vector $\bar{c}_n$ and the mean Euclidean distance σ_D to $\bar{c}_n$. These quantities form a background model, and a newly observed ILHS (y_o, h_o, s_o) is classified as foreground if:

$$|y_o - \mu_y| > \alpha\sigma_y \ \vee \ \|\bar{c}_n - s_o h_o\| > \alpha\sigma_D,$$

where α is the threshold. Then, if a pixel is classified as foreground by fulfilling the above conditions, the pixel is checked if it is within shaded background as follows:

$$\begin{aligned} y_o < \mu_y \wedge |y_o - \mu_y| &< \beta\mu_y, \\ s_o - \bar{R}_n &< \tau_{ds}, \\ \left\| h_o \bar{R}_n - \bar{c}_n \right\| &< \tau_h, \end{aligned}$$

where, $\bar{R}_n = \|\bar{c}_n\|$. They defined performance measures, DR, FR, MP, RM, and QMS, for motion segmentation. They compared their method with RGB+NRGB, NRGB+NRGB, and RGB+HSV. Their method outperforms the others. This method is relatively easy to implement and shows good performance of background subtraction.

1.7 Depth Segmentation

Disparities between stereo or multi-view images can be found by different methods. Among the many methods, recent energy minimization type methods have been reported to produce high quality disparity maps. Belief propagation [55] and Graph-cut are often employed as optimization tools. Meltzer *et al*. proposed an improved belief propagation, namely the tree re-weighted belief propagation [56].

Biologically inspired methods make one category of computational stereo disparity methods. The cooperative method is one of the classic biologically inspired methods, which has originally been proposed by Mar and Poggio [57] for computing stereo disparity and has been extended for occlusion detection by Zitnick and Kanade [58]. They propose restricting the matching values in iterations depending on the image similarity between the pixels in the two images corresponding to a disparity value to prevent over-smoothing. Occlusions are detected by examining the magnitude of the matching score with a specific threshold. The cooperative methods use the matching score volume represented by network status for computing the disparity map. However, it is not well known what the global measure in this optimization is, and what cost function is to be minimized in this process. Recently, Brockers [59] presented a cooperative stereo matching algorithm with explicit energy minimization, where the matching score volume consistent with initial matching scores was smoothed. He introduced a color-based adaptive local support, which is a factor of smoothness between pixels in the energy function. Occlusion detection and post-processing to find sub-pixel level disparity maps were also proposed in this method. Energy minimizations using a global optimization method have recently become mainstream in approaches of stereo depth computation.

Over-segmentation using so-called super-pixels, can be combined with existing methods. Zhang and Kambhamettu [60] introduced the segmentation-based approach into the cooperative algorithm. Their algorithm limits the local support area within the color segment in order to preserve boundaries. However, they added neighboring segments into the support area, because disparity propagation within only one segment was not sufficient. This, however, conflicts with the concept of preserving boundaries.

Woo *et al.* [30] proposed an algorithm to segment a foreground object using depth. They formulate the segmentation as a MAP estimation, where the probability of segmentation, disparity, disparity contour, intensity contour, and smooth stereo image pairs is defined based on the MRF/GRF (Gibbs random fields) framework. The MAP estimation is broken down into three components: estimation of the smooth stereo image pair and intensity contour given a stereo image pair, estimation of disparity map and disparity contour given the smooth stereo image pair and intensity contour, and estimation of segmentation given the disparity map and disparity contour. The initial intensity contour is given. The initial disparity map is computed using the hierarchical OBM (overlapped block matching) method. The initial segmentation is computed by background subtraction. The experimental result looks good, but only one result has been shown. Since this method uses a background image to obtain an initial guess of the segmentation, its application is limited.

1.8 Cue Integration

One type of visual cue is often not sufficient to produce good results for image segmentation. However, it is not straightforward how to integrate different types of visual cues. The following literature proposed segmentation utilizing multiple visual cues.

Alpert *et al.* [25] proposed an image segmentation algorithm based on intensity and texture. The probability that a pair of regions are in the same segment is defined as $P\left(s_{ij}^{+}|\overrightarrow{\mathcal{H}}_i, \overrightarrow{\mathcal{H}}_j\right)$, where s_{ij} is a binary random variable taking values s_{ij}^{+} or s_{ij}^{-}, where s_{ij}^{+} indicates that two region i and j are in the same segment. $\overrightarrow{\mathcal{H}}_i$ is a set of observations which includes average intensity and edge filter responses in the region i. A graph is constructed, where regions are nodes and edges are associated with a weight $p_{ij} = P\left(s_{ij}^{+}|\overrightarrow{\mathcal{H}}_i, \overrightarrow{\mathcal{H}}_j\right)$. The graph is progressively shrunk by coarsening. Experimental results show that the proposed method is the best in the *F-measure* compared to SWA, N-cuts and Mean-Shift. The experiments also showed that the proposed algorithm covers a foreground object with fewer number of fragments compared with these other methods.

Kohli *et al.* [43] proposed an image segmentation algorithm using a CRF model with unary, pairwise and higher order potentials. The unary potentials are a combination of color, texture, location, and shape. The pairwise potential is based on the difference in color of neighboring pixels. The higher order potentials are obtained using multiple mean-shift segmentations with different parameters. They claim better segmentation results using a robust P^n Potts model instead of a P^n Potts model. Experiments shows that accurate segmentations are obtained.

Torralba [61] proposed a learning algorithm for detecting a large number of object classes in cluttered scenes. Their method shares features across object classes, and the detectors for each class are trained jointly. To obtain the classifier they use a joint boosting algorithm. They also provide an efficient procedure for its implementation because this method need to search among all feature, combinations of classes, and thresholds.

Chapter 2

Analysis of Torque Operator

2.1 Definition

Torque is a measure used in physics to express how much a force acting on an object causes the object to rotate. It is defined as:

$$\vec{\tau} = \vec{r} \times \vec{F}, \tag{2.1}$$

where $\vec{\tau}$ is the torque vector. $\vec{r}$ is the displacement vector from the axis of rotation to the point where the force is applied. $\vec{F}$ is the force vector. Based on this concept we define a torque measure for images. It is treated as if a force is applied at a point on an image edge along the tangent of the edge. For an arbitrary point, which we call the center point, the rotation axis for the torque is imagined as if it is going through the image point in three-dimensional space. Then, we can measure the torque at any point in the image with respect to the rotation axis as it is defined in physics. Figure 2.1 illustrates the concept of the torque in an image. Since the displacement vector and the force vector are on the image surface, the torque vector is always perpendicular to the image surface. Therefore, we call the value of the torque vector along the axis perpendicular to the image simply the *torque* or *torque value* here and after.

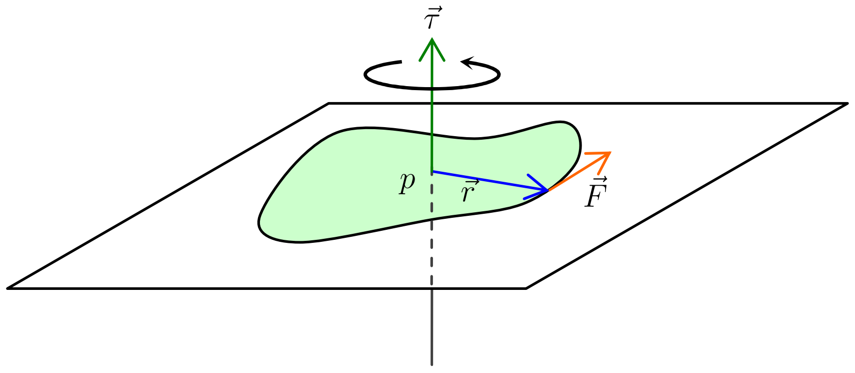


Figure 2.1: Torque for Images. Consider a center point p and an edge point q, to which we assign a force vector $\vec{F}$ along the tangent of the edge. Denoting as $\vec{r}$ the vector from p to q, the torque vector at p is defined as $\vec{\tau} = \vec{r} \times \vec{F}$. Its value along the axis perpendicular the image will be called the *torque*.

2.1.1 Continuous Edge Curve

In general, an image edge is represented by a function $C \in \mathbb{R}^2$ with parameter t as follows:

$$C\left(t\right) = \left(C_x\left(t\right), C_y\left(t\right)\right)^T. \tag{2.2}$$

The derivative of C is treated as the force applied at an edge point. Thus we can define the *torque for points on edge curves* as:

Definition 2.1.1 (Torque for points on edge curves)

$$\tau_p\left(t\right) = \left(C\left(t\right) - p\right) \times C^{'}\left(t\right), \tag{2.3}$$

where p is the point where the rotation axis pierces the image. For simplicity, the cross product for two dimensional vectors is obtained by cross-multiplying the vectors, which is equivalent to considering the third component of the cross product *of three dimensional vectors, whose first two components are the two dimensional vectors and whose third element is zero.*

Figure 2.2 gives an illustration of the geometric relation of the two vectors in eq. (2.3).

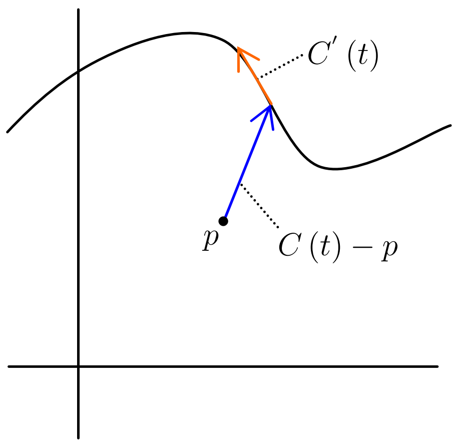


Figure 2.2: Torque for Points on Edge Curves.

Based on the definition of the torque for a point we define the *torque of an image patch* as the sum over the torque values of all edge points in the image patch. A rigorous definition of the torque of an image patch of arbitrary shape is as follows:

Definition 2.1.2 (Torque of an image patch (continuous))

$$\tau_P = \sum_i^N \sum_j^{M^i} \frac{1}{2\,|P|} \int_{t_0^{i,j}}^{t_1^{i,j}} \tau_p^i(t)\,dt, \tag{2.4}$$

where P denotes an image patch, and p is a point inside the patch. $|P|$ is the area of the patch. N is number of edge curves overlapping the image patch. $\tau_p^i(t)$ is the torque at value t of the i-th edge curve. The i-th edge curve has M^i segments within the image patch, and the j-th segment of the i-th edge curve is represented between

$t_0^{i,j}$ and $t_1^{i,j}$ in parameter t.

In this definition, the sum of the torque value for each edge segment is normalized by $1/(2|P|)$ in order to make the torque of an image patch invariant to scale. The reason for using the factor 2 in the denominator is explained in the section 2.2.2.

2.1.2 Discrete Edge Points

Since our images are discrete, edges are represented by a set of pixels in the image instead of continuous edge curve. Then, the force at a edge pixel is defined by an oriented unit vector in the direction of the edge, and the *torque value of an edge pixel* amounts to:

Definition 2.1.3 (torque value of an edge pixel)

$$\tau_{pq} = \|\vec{r}_{pq}\| \sin\theta_{pq}, \tag{2.5}$$

where p is the point on the image through which the rotation axis passes. q is an edge pixel. $\vec{r}_{pq}$ is the displacement vector from p to q, and θ is the angle between the displacement vector and the edge orientation.

Note that in our definition edges are oriented. Thus, the value of the torque can have positive and negative values. If multiple images are available we can define the orientation on the basis of depth. In the case of single images we define it as perpendicular clockwise to the image gradient, such that the brighter side is on its right and the darker side on its left.

Similarly to the definition on edge curves, the discrete *torque of an image patch* is defined as:

Definition 2.1.4 (torque for an image patch (discrete))

$$\tau_P = \frac{1}{2\,|P|} \sum_{q \in E(P)} \tau_{pq}, \tag{2.6}$$

where $E\,(P)$ is a set of edge pixels in the patch P, and p is a point inside the patch for the rotation axis.

In principle the shape of the patch could be arbitrary, but in this thesis, we examine disc, square, rectangle, and elliptic patches, where p is at the center of the patch. For the case of disk or square patch, it is specified by its center point (x, y) and its scale, *i.e.* radius or width, s. Therefore, the torque of a patch is specified as a value in three-dimensional space (x, y, s):

$$\tau\,(x, y, s)\,. \tag{2.7}$$

We call this set of torque values in three-dimensional space as *torque volume.*

2.2 Fundamental Properties

In this section, some of the basic properties of the torque are treated formally. Specifically, we start with a discussion on the superposition principle and the relationship between torque and area. Next, an intuition for torque is given by analyzing the torque on simple shapes. Then the properties of torque extrema observed over scale are discussed. Torque maps for multiple objects in different configurations are shown, and lastly the similarity of the torque to the medial axis transform is discussed.

2.2.1 Principle of Superposition

From the definition 2.1.2, the torque of an image patch is decomposed into the torque values of the edge curves or points in the patch. This fact can be stated as follows:

Lemma 2.2.1 (Principle of Superposition) *Let's say that the edge curves overlapping with a given patch are represented as the union of edge curves as follows:*

$$C = \bigcup_i \left\{ C_i \left(t_i \right) | t_0 \leq t_i \leq t_1 \right\}. \tag{2.8}$$

Then, the torque for C is the superposition of torques for C_i.

Similarly, the lemma can be also derived from the definition 2.1.4 by replacing C_i by the torque value for each edge pixel, τ_{pq}.

2.2.2 Torque and Area

Since the torque is defined by the cross product of vectors, it is essentially related to the area defined by these vectors as shown in Fig. 2.3. This relationship can easily be extended to edge curves. Assuming edges are clean continuous curves, the amount of the torque of a patch is related to the position of the curves in the patch and their shape. The torque of a closed curve, completely inside the patch, is proportional to its area. The closer the patch boundaries surround the curve, the larger the torque value becomes. For curve segments intersecting the boundary of the patch with center p at two intersection points q_1 and q_2, the torque is proportional to the area enclosed by the edge curve and the two line segments pq_1 and pq_2. This is depicted in Fig. 2.4.

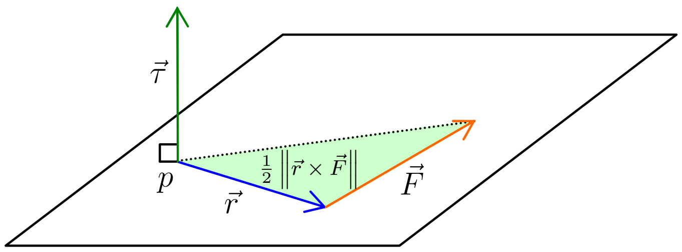


Figure 2.3: Cross Product and Area. The triangle enclosed by the vector $\vec{r}$ and $\vec{F}$ is equivalent to $\left\|\vec{r}\times\vec{F}\right\|/2$.

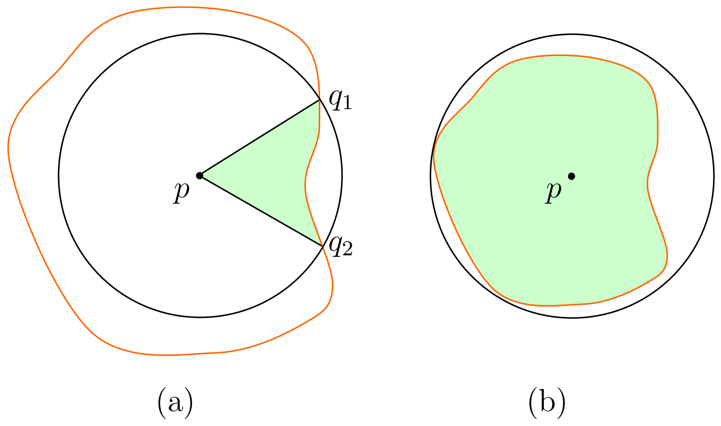


Figure 2.4: Relationship between Torque and Area. (a) The disk patch is smaller than the object, and it intersects only a part of the boundary. (b) The disk patch covers the whole object boundary. In this figure the patches are drawn of same size, and the object size is changed from (a) to (b) to visualize the normalization effect in the *torque of a image patch*, which is proportional to the hatched area.

For a more precise analysis, the relationship between torque and area is stated as follows:

Lemma 2.2.2 (Relationship between Torque and Area) *The absolute value of the torque of a patch without normalization is equivalent to twice the area enclosed by segments of edge curves overlapping the patch and line segments connecting the point of the rotation axis and the end points of the edge curve segments* (i.e. *the hatched area in Fig. 2.4*).

Proof (2.2.2) *Let's say that* $\left\{C(t) = (C_x(t), C_y(t))^T : t0 \leq t \leq t1\right\}$ *is an overlapping segment of an edge curve* $C(t)$ *on the patch. The point of rotation axis is denoted by* p. *The area enclosed by the segment of the edge curve and the two line segments* $C(t_0) - p$ *and* $C(t_1) - p$ (i.e. *the hatched area in Fig. 2.5*) *can be computed as follows:*

$$\begin{aligned} A &= \frac{1}{2}\int_{t_0}^{t_1} \|C(t) - p\| \cdot \frac{(-C_y(t) + y_p, C_x(t) - x_p)}{\|C(t) - p\|} \cdot C'(t)\,dt \\ &= \frac{1}{2}\int_{t_0}^{t_1} (C(t) - p) \times C'(t)\,dt. \end{aligned} \tag{2.9}$$

On the other hand, the torque of a point $p = (x_p, y_p)$ *without normalization is written as follows:*

$$\tau = \int_{t_0}^{t_1} (C(t) - p) \times C'(t)\,dt. \tag{2.10}$$

Fig. 2.6 illustrates the geometric interpretation of the term inside integral in eq. (2.9). At a single point we have an an infinitesimal triangular area, and the integral between t_0 *and* t_1 *is equivalent to the area specified in Fig. 2.5. Therefore, the torque without normalization is equivalent to twice of the area.* ■

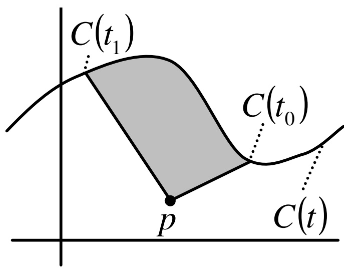


Figure 2.5: Area of Curve with regard to Point.

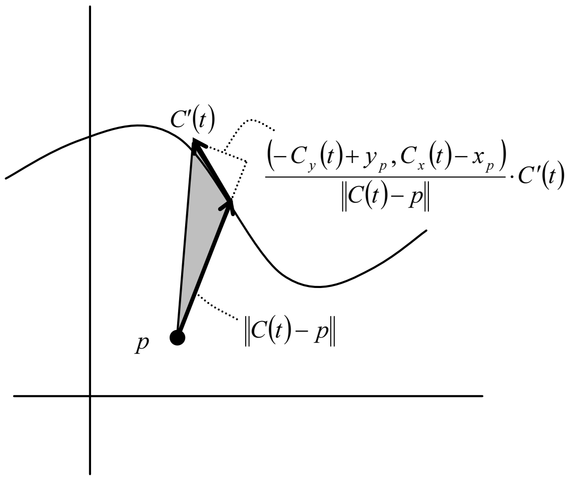


Figure 2.6: Area for Infinitesimal Curve Segment.

There is an another way to verify the relationship between torque and area. The torque for the edge curve $C(t)$ with regard to p can be rewritten as follows:

$$\begin{aligned}
\tau &= \int_{t_0}^{t_1} (C(t) - p) \times C'(t)\, dt \\
&= \int_{t_0}^{t_1} \left\{ \begin{pmatrix} C_x(t) \\ C_y(t) \end{pmatrix} - \begin{pmatrix} x_p \\ y_p \end{pmatrix} \right\} \times \begin{pmatrix} C'_x(t) \\ C'_y(t) \end{pmatrix} dt \\
&= \int_{t_0}^{t_1} (C_x(t) - x_p)\, C'_y(t) - (C_y(t) - x_p)\, C'_x(t)\, dt \\
&= [C_x(t)\, C_y(t)]_{t_0}^{t_1} - \int_{t_0}^{t_1} C'_x(t)\, C_y(t)\, dt - x_p\, [C_y(t)]_{t_0}^{t_1} \\
&- [C_y(t)\, C_x(t)]_{t_0}^{t_1} - \int_{t_0}^{t_1} C'_y(t)\, C_x(t)\, dt - y_p\, [C_x(t)]_{t_0}^{t_1} \\
&= \int_{t_0}^{t_1} C'_y(t)\, C_x(t) - C'_x(t)\, C_y(t)\, dt \\
&+ y_p\, (C_x(t_1) - C_x(t_0)) - x_p\, (C_y(t_1) - C_y(t_0)) . \qquad (2.11)
\end{aligned}$$

Without loss of generality, we can take the origin of the coordinate system as (x_p, y_p). Then, the above equation can simply be written as follows:

$$\begin{aligned}
\tau &= \int_{t_0}^{t_1} C(t) \times C'(t)\, dt \\
&= \int_{t_0}^{t_1} C'_y(t)\, C_x(t) - C'_x(t)\, C_y(t)\, dt. \qquad (2.12)
\end{aligned}$$

The above equation can again be rewritten as :

$$\begin{aligned}
\tau &= C_x(t_1)\, C_y(t_1) - C_x(t_0)\, C_y(t_0) - 2\int_{t_0}^{t_1} C'_x(t)\, C_y(t)\, dt \\
&= 2\int_{t_0}^{t_1} C'_y(t)\, C_x(t)\, dt - C_x(t_1)\, C_y(t_1) + C_x(t_0)\, C_y(t_0) . \qquad (2.13)
\end{aligned}$$

Referring the Fig. 2.7, the above equation can be interpreted as follows: The area hatched with diagonal lines equals to the area of the big triangle $(\frac{1}{2} C_x(t_1)\, C_y(t_1))$

minus the small triangle hatched with dots ($\frac{1}{2}C_x(t_0)\,C_y(t_0)$) minus the area hatched with waves ($\int_{t_0}^{t_1} C_x'(t)\,C_y(t)\,dt$).

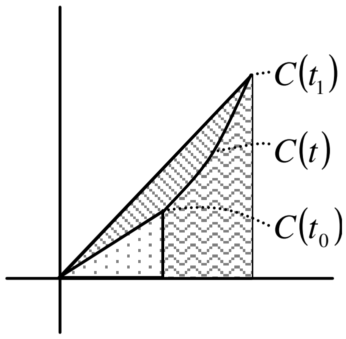


Figure 2.7: Another Geometric Interpretation.

2.2.3 Torque for Simple Shapes

First, we start with examining the torque for a line segment. Many simple shapes can be drawn with a set of line segments as polygons. By the definition in eq. 2.1.2, the torque of a patch is the sum of the torque values of all edge curve segments inside the patch as stated in lemma 2.2.1. Therefore, the torque for a polygon can be decomposed into the torque values of line segments. A line segment is specified by its end points $p_s = (x_s, y_s)^T$ and $p_e = (x_e, y_e)^T$. The line segment is written as a function of parameter t as follows:

$$\begin{aligned} C(t) &= p_s + t\,(p_e - p_s) \\ &= (x_s + t\,(x_e - x_s)\,, y_s + t\,(y_e - y_s))^T\,. \end{aligned} \tag{2.14}$$

We assume that the line segment overlaps with the patch within $t_0 \le t \le t_1$. The integral of the torque in eq. (2.3) with eq. (2.14) is solved analytically between t_0

and t_1 as follows:

$$\begin{aligned}\tau_p &= (p_s \times p + p_s \times p_e + p_e \times p)\,(t_1 - t_0) \\ &= (x_s y_p - y_s x_p + x_s y_e - y_s x_e + x_e y_p - y_e x_p) \cdot (t_1 - t_0)\,. \end{aligned} \tag{2.15}$$

A disk patch overlapping a line segment it is depicted in Fig. 2.8.

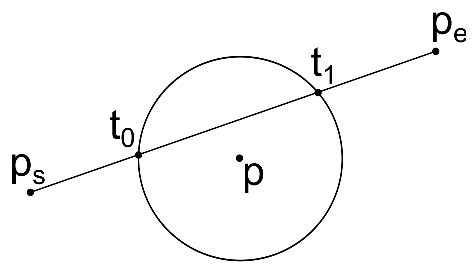


Figure 2.8: Torque of Line Segment for Disk Patch

A torque map is computed by computing the torque of a patch at every location in the image. For example, if the patch is a disk of specific radius, the torque map is computed by applying the disk patch at every point in the image. A torque map for a line segment is shown in Fig. 2.9.

When we use disk patches, computing the torque map for rotated images is equivalent to rotating the torque map. A polygon can be seen as consisting of the rotated and translated copies of a line segment. Therefore, the torque map for a polygon can be produced by taking the sum of rotated and translated copies of the torque map a line segment according to the principle of superposition. Figure 2.10 is an example to show this principle. Figure 2.10(a) shows an image consisting of a line segment and its rotated and translated copy . Figure 2.10(b) shows the torque map for the line segment. Figure 2.10(c) shows the torque map rotated and translated

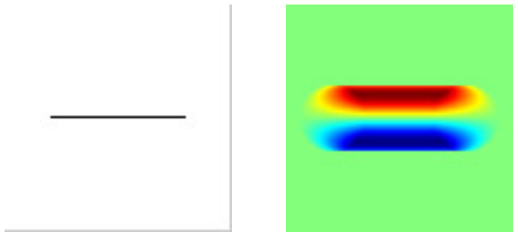

(a) (b)

Figure 2.9: Torque of Line Segment using Disk Patches. (a) shows an image of a line segment. (b) shows the color coded torque map. The torque value is color coded so that warm colors indicate positive value and cool colors indicate negative value.

in the same way as the line segment. Figure 2.10(d) shows the torque map for the two line segments, which is equivalent to the sum of Fig. 2.10(b) and (c).

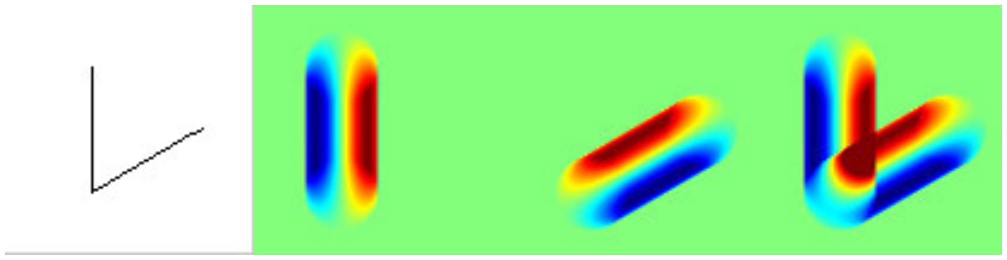

(a) (b) (c) (d)

Figure 2.10: Superposition of Torque Maps. (a) shows an image consisting of a line segment and its rotated and translated copy. (b) shows the torque map for one line segment. (c) shows the torque map rotated and translated in the same way as the line segment. (d) shows the torque map for the two line segments, which is equivalent to the sum of (b) and (c).

The torque map changes with the size of the patch in addition to the shape of the patch. Examples of torque maps using different patch sizes are shown in Fig. 2.11

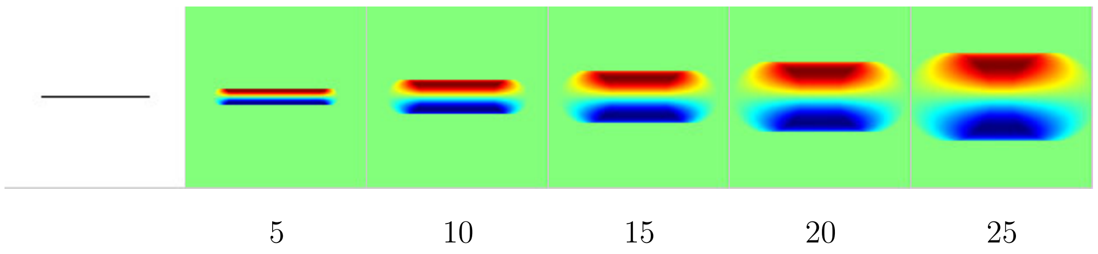


Figure 2.11: Torque Maps for Line Segment using Disk Patches of Different Size. The numbers below the torque maps denote the radius of the disk patch in pixels. The image is size of 101 × 101 pixels.

2.2.4 Torque Extrema

According to the principle of superposition, multiple line segments produce multiple torque distributions, and these distributions are added when they collide. The torque of a patch by definition is expected to be large in magnitude when the edges within a patch are surrounding the center point in a way that their forces add up, while it is expected to be small in magnitude when the edges are few or random. Therefore, it is expected that the torque measure is useful for finding the location and the scale, where edges are in structure of their surrounding. Figure 2.12 shows an example, where the structured edges form a triangle. Observing the torque over different patch sizes, we see that the peak of the torque indicates the center of the triangle and the corresponding patch size indicates the size of the triangle. Figure 2.13 gives another example, where the structured edges form a circle. Square patches were used in the computation. Like in the above example, the peak in the torque values over scale at the center of the circle indicates the size of the circle. Furthermore, the maximum torque value represents how well the edges surrounding the point are aligned. The patch containing the maximum torque value indicates the scale of the best aligned edges (within the range of patch sizes considered). The torque tends to be of same sign inside close edges, and of opposite sign outside. The sign depends on whether the edge direction is clock-wise or counter-close-wise.

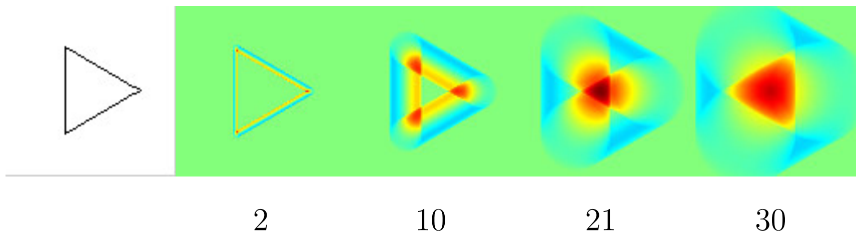


Figure 2.12: Torque Maps for a Triangle over Patch Sizes. The numbers below the torque maps denote the radius of the disk patch in pixels. The image is size of 101×101 pixels, and the height of the triangle is 40 pixels. The radius of the inscribed circle of the triangle is 20 pixels.

Regarding extrema of the torque, a theoretical analysis using simple linear edge can give us an understanding of the basic factors determining the value of the torque. As we see from the relationship between torque and area (sec. 2.2.2), the torque for an edge line in a disk patch is proportional to the area of the triangle formed by the lines from center of patch to the intersection points, and the edge segment, as depicted in Fig 2.14 Referring to the figure, the orange lines represent edges, and the green triangles are proportional to the torque value. Taking the normalization factor ($2\pi r^2$) into account, the torque for a linear edge is written as a function of the radius of the disk patch r and the distance, d from the center of the disk to the edge, i.e.

$$\tau = \pm \frac{d\sqrt{r^2 - d^2}}{\pi r^2}. \tag{2.16}$$

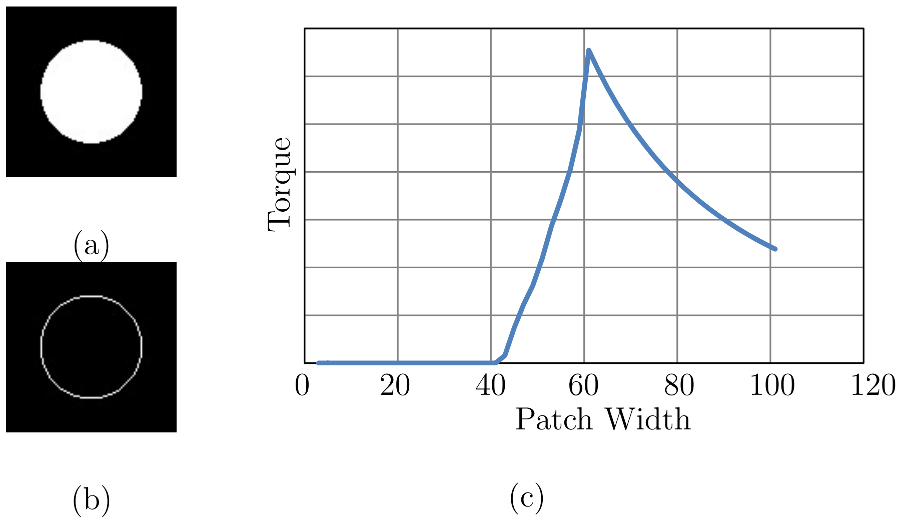


Figure 2.13: Torque Maps for a Circle over Patch Sizes. (a) is the image, and (b) is the edge map of the image. The circle is located in the middle of the image, and its diameter is 61 pixels. The torque values are computed at the center point of the image over different patch widths. Plot (c) shows, that the maximum torque at the center of image is obtained when the patch size fits the circle.

This function is plotted over r in Fig 2.14. The graph can also can be interpreted as a plot over $1/d$ because changing r to αr is equivalent to changing d to $\frac{d}{\alpha}$ in eq. (2.16). The lower part of this figure visualizes the relationship between the disk patch, the linear edge and its torque value. As r increases, the area of the triangle increases, however it shrinks when we consider the normalization to a unit disk patch. Considering the normalization factor the torque value is proportional to the area of the triangle shown at the bottom of the figure. Drawings in the same column correspond to the same torque value plotted in the graph.

The derivative of τ with respect r amounts to:

$$\frac{\partial \tau}{\partial r} = \pm \frac{d\left(2d^2 - r^2\right)}{\pi r^3 \sqrt{r^2 - d^2}}. \tag{2.17}$$

The derivative is equal to zero when $r = \sqrt{2} \cdot d$, and the torque at this radius has an extremum of value $\tau = \pm 1/(2\pi)$. The torque maps for a line segment at different patch sizes are shown in Fig. 2.11. The extrema of torque is at a distance $d = r/\sqrt{2}$ for a specific patch size, or at a patch size of $r = \sqrt{2} \cdot d$ for a specific point when the distance to the line segment is d.

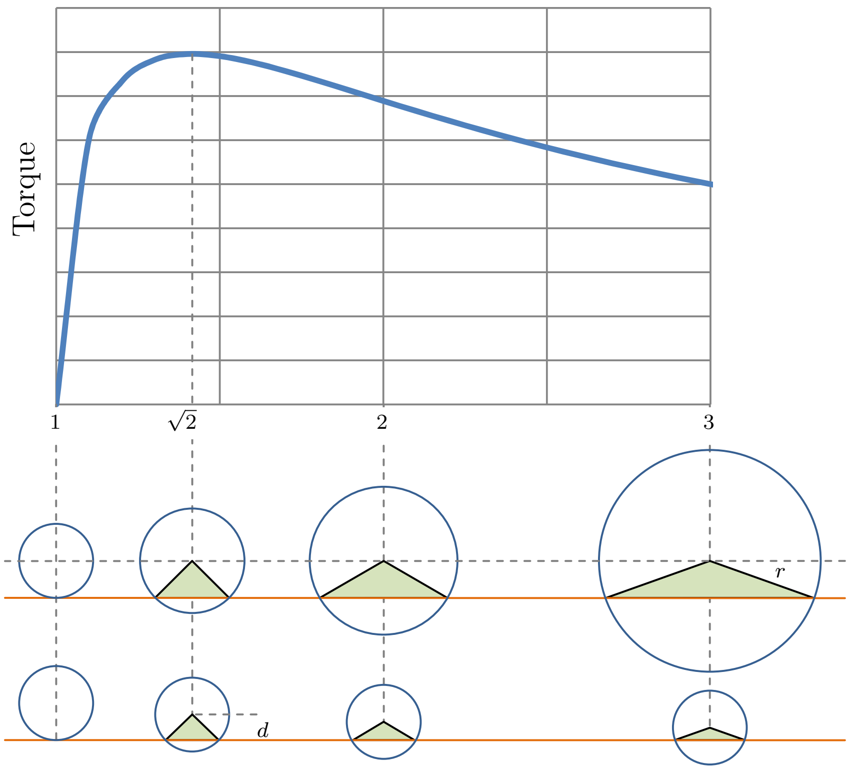


Figure 2.14: Torque Transition over different Patch Sizes

2.2.5 Torque Value Map and Scale Map

We define a data-structure that combines the torque values for different patch sizes, which we can describe by the following equations:

$$V(x,y) = \tau(x,y,\hat{s}(x,y)), \tag{2.18}$$

$$S(x,y) = \operatorname{sgn}(V(x,y)) \cdot \hat{s}(x,y), \tag{2.19}$$

$$\hat{s}(x,y) = \underset{s}{\operatorname{argmax}} \left|\tau(x,y,s)\right|, \tag{2.20}$$

where $\tau(x,y,s)$ is the torque value at point (x,y) with patch size (scale) s. We call this three-dimensional volume of τ the *torque volume*, and V and S the *torque value map* and *scale map*, respectively. Figure 2.15 shows examples of torque value maps and scale maps for some simple figures, where the darker red color indicates larger positive value and the darker blue color indicates smaller negative value. The edge maps are overlaid in black in the figure. In the cases of these simple shapes, the torque value map directly provides the location of the object, *i.e.* the center of the polygons, and the scale map provides its scale (at the point corresponding to the extrema in the value map).

2.2.6 Texture vs Boundary

The torque will be larger when it contains extended contours and will be largest if the contours are closed. On the other hand, the torque will be small if the edges are random. Thus, intuitively texture edges lead to small torque value and boundary edges to large values. Figure 2.16 is an explanatory example to demonstrate that the torque operator captures closedness of boundary edges being

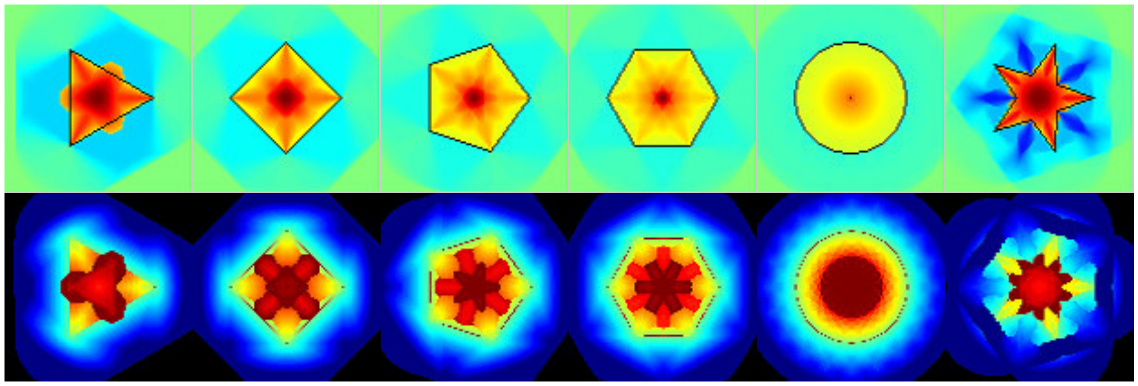


Figure 2.15: Torque Value Maps and Scale Maps for Simple Shapes. Upper row shows torque value map for each shape. Edges are drawn onto the torque value map with black lines. The corresponding scale maps are shown in the lower row.

little effected by random texture edges. In this example, an image of a textured circle on white background was generated. Although some small effects of texture edges are observed in the torque value map, the strongest extrema stays at the center of the circle, and the the region of same sign of torque roughly represents the object region. Although the darkest region inside the circle has a closed boundary, and the surrounding regions are all brighter than the dark region, there does not appear a strong extrema in the torque for the dark region.

2.2.7 Different Configurations

Torque value maps and scale maps are more complex for multiple objects located close to each other or for objects of different topology. Figure 2.17 shows two examples of theoretical and empirical value maps and scale maps for different configurations; one is for the case of one object on top of another object, and the

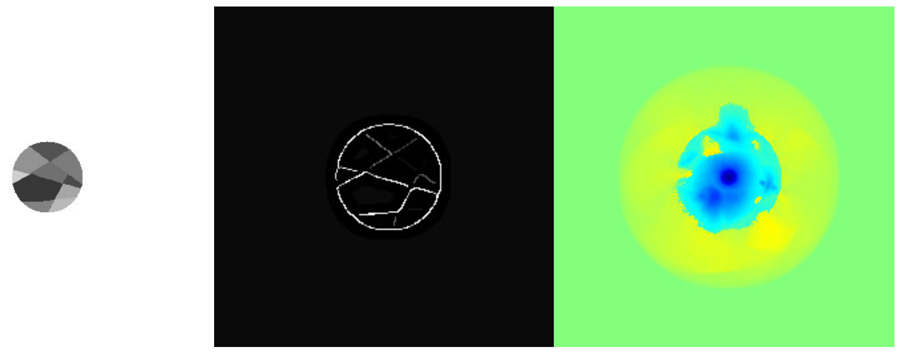

(a) (b) (c)

Figure 2.16: Example Torque Map for Textured Object: (a) Synthesized textured object image. (b) Edge map. (c) Torque value map.

other is for the case of an object with a hole. The edge direction is defined such that the darker region is on the left side of the edge along its direction. For the case of two objects on bright background, and the brighter object on top of the darker object, both the theoretical and the empirical torque value map have negative torque for the darker object, but positive torque values appear in the brighter object region around the edges shared with the darker object. For the darker object, the edges are closed, because the torque definition takes the the direction into account. On the other hand, the edges at the border between the brighter object and the darker object conflict with other edges of the brighter object in direction. These edges give torque values of opposite sign, and result in the positive torque value region close to the edge. For the case where an object has a hole, the edges of the outer and inner boundaries provide torque values of the same sign for the object region, and the edges of the inner boundaries provide a torque of the opposite sign for the hole. In these examples, the object has negative torque, and positive torque appears at the region of the hole.

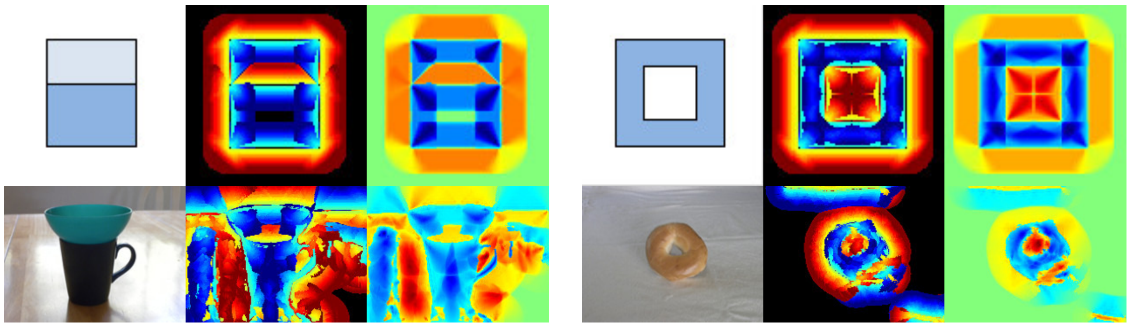


Figure 2.17: Torque for Different Configuration. The first and the second row show the theoretical and empirical torque value and scale maps, respectively. The first and the fourth column show the test images. The second and the fifth column show scale maps. The third and sixth show torque value maps.

2.2.8 Two Circles Configurations

There is an infinite number of configurations of two objects, because the shapes of objects are arbitrary. Using topology is a way to abstract away shape differences. However, the torque is strongly affected by the shape. The shape of an apple and the shape of a snake are topologically same, but the torque maps for these objects are different. The shape of the patches used has an influence on the torque. For example, for the case of disk patches, the boundary of an apple is well suited for the patch, but the boundary of a snake is not. Thus, the choice of the patch shape is essentially an assumption on the desired or most probable shape of objects. In order to make the analysis of different configurations meaningful, we will consider disk patches, and the object are circles. The configuration of two circles can be defined by the relative radius and the relative location of one circle to a unit circle.

What we want to know, is the border where catastrophic changes in the torque map occur.

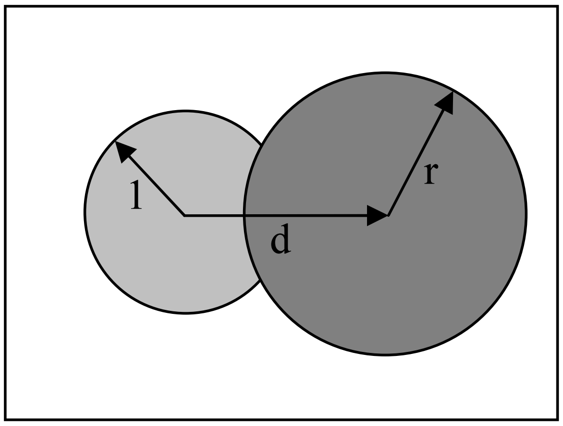


Figure 2.18: Two Circles Configuration Definition.

As d and r change, as shown in figure (a)-(f) below, catastrophic changes in the torque are expected at (b), (d), (f).

In this experiment, we used a white (brightness=1) background and dark circles. The near circle was filled in black (brightness=0), and the far circle was filled in gray (brightness=0.5). The far circle is drawn with 20 pixel radius, and here r and d are in relative scale as the far circle size is normalized to 1. The following figures show torque value maps over changes in r and d. Edges are overlaid to the torque value map in black. As can be seen, the generated torque value maps well indicate object location. Significant torque sign changes by the effect of occluding object are not observed.

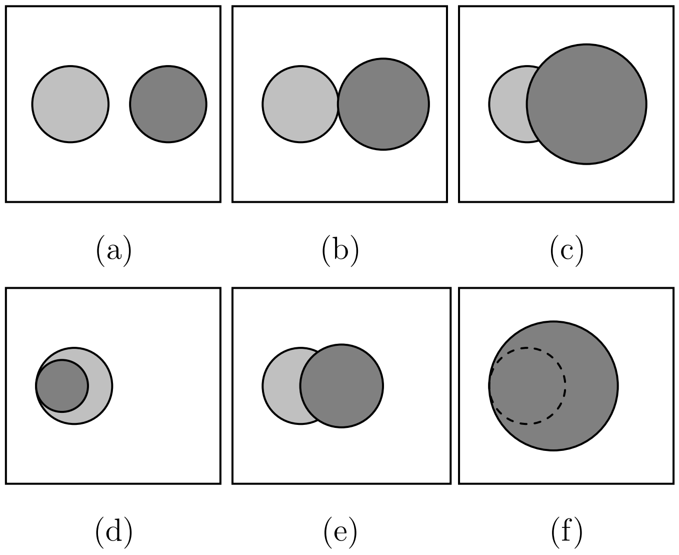


Figure 2.19: Two Circles Configuration Samples.

2.2.9 Torque based on Gradient

Up to now, the torque has been defined on edges. Images have strong edges and weak edges, but the strength of edges has not been considered in the definition. Edges are normally detected based on image gradients, and the gradient carries the strength of edges. Next we define a variation of the torque defined on image gradients. The *torque on gradients* is defined as follows:

Definition 2.2.1 (Torque on gradients)

$$\tau_{pq} = \vec{r}_{pq} \times \tilde{\nabla} I\left(q\right), \tag{2.21}$$

where p is the point, where the rotation axis pierces the image. q is a point on the image. r_{pq} is the displacement vector from p to q. $\tilde{\nabla} I\left(q\right)$ is the 90 degree rotated gradient of the image I at q representing an oriented edge-like vector, which

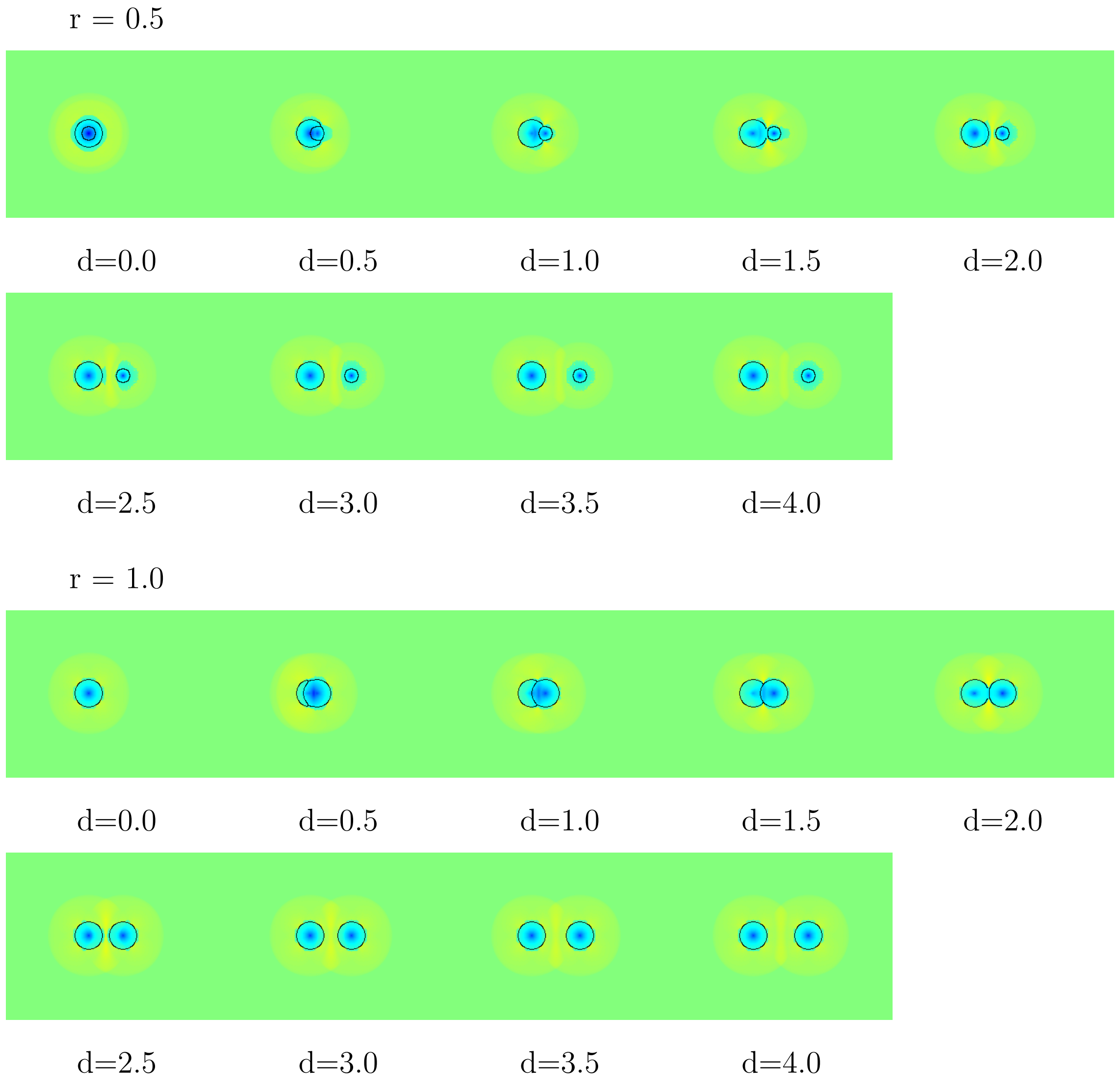


Figure 2.20: Two Circles Configurations Experiment 1/2.

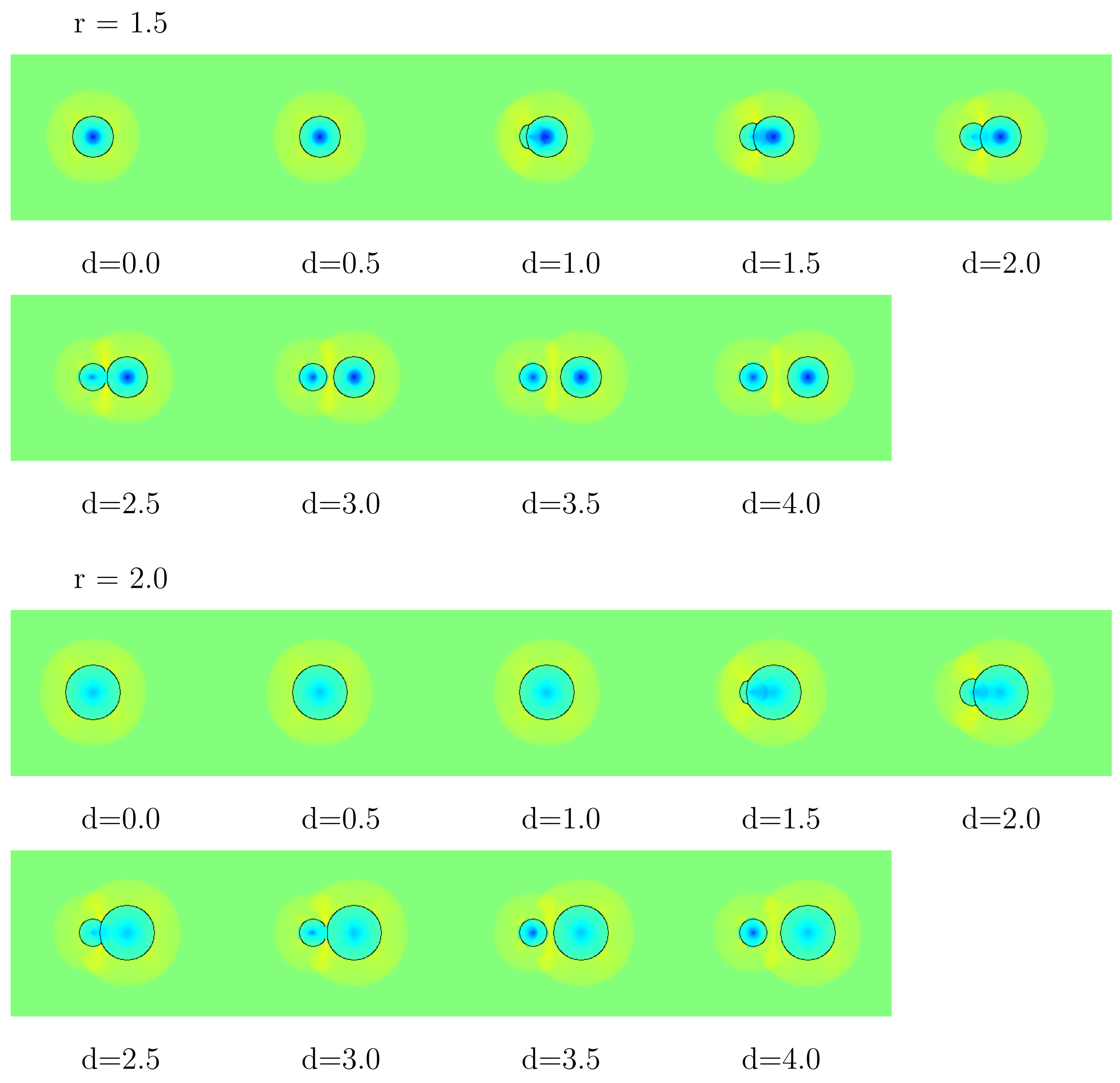


Figure 2.21: Two Circles Configurations Experiment 2/2.

is defined as $\left(\frac{\partial I}{\partial y}, -\frac{\partial I}{\partial x}\right)$.

Similarly, the *Torque on gradients for a patch* is defined as follows:

Definition 2.2.2 (Torque on gradients for a patch)

$$\tau_P = \frac{1}{2\,|P|} \int_{q \in P} \tau_{pq} dq, \tag{2.22}$$

where P is the patch, and the integral is taken over the patch.

Next we show that the torque of gradients for a disk patch is equivalent to the difference of the average image intensity inside the patch and the average image intensity on the patch boundary.

Lemma 2.2.3 (Torque of Gradients and Average Intensity) *The torque based on image gradients is equivalent to the average intensity over the image patch minus the average intensity on the boundary of the patch. Using disk patches this amounts to:*

$$\begin{aligned}\tau_P = &\frac{1}{\pi R^2} \int_0^R \int_{-\pi}^{\pi} I\left(r, \theta\right) \cdot r d\theta dr \\ &- \frac{1}{2\pi R} \int_{-\pi}^{\pi} I\left(R, \theta\right) \cdot R d\theta,\end{aligned} \tag{2.23}$$

where $I\left(r, \theta\right)$ is image intensity at $\left(r, \theta\right)$ in the polar coordinate system.

Without loss of generality, it is assumes that the center of the patch is the origin of the coordinate system so that $\vec{r}_{pq} = (x, y)$ for $q = (x, y)$. Then, eq. (2.23) can be derived as follows using $(x, y) = (r\cos\theta, r\sin\theta)$:

Proof (2.2.3)

$$\begin{aligned}
\tau_P &= \frac{1}{2\pi R^2}\int_0^R\int_{-\pi}^{\pi}\left\{x\left(-\frac{\partial I}{\partial x}\right)-y\frac{\partial I}{\partial y}\right\}rd\theta dr\\
&= \frac{1}{2\pi R^2}\int_0^R\int_{-\pi}^{\pi}\cos\theta\left(\sin\theta\frac{\partial I}{\partial\theta}-r\cos\theta\frac{\partial I}{\partial r}\right)\\
&-\sin\theta\left(\cos\theta\frac{\partial I}{\partial\theta}+r\sin\theta\frac{\partial I}{\partial r}\right)rd\theta dr\\
&= -\frac{1}{2\pi R^2}\int_0^R\int_{-\pi}^{\pi}r^2\frac{\partial I}{\partial r}d\theta dr\\
&= -\frac{1}{2\pi R^2}\int_{-\pi}^{\pi}\left\{R^2I\left(R,\theta\right)-\int_0^R 2rI\left(r,\theta\right)dr\right\}d\theta\\
&= \frac{1}{\pi R^2}\int_0^R\int_{-\pi}^{\pi}I\left(r,\theta\right)\cdot rd\theta dr\\
&-\frac{1}{2\pi R}\int_{-\pi}^{\pi}I\left(R,\theta\right)\cdot Rd\theta. \qquad (2.24)
\end{aligned}$$

The equation above is equivalent to eq. (2.23) ∎

Lemma 2.2.3 has been given for a disk patch, but it is also holds for square patch, as shown next.

$$
\begin{aligned}
\tau\left(x_0, y_0\right) &= \int_{y_t}^{y_b} \int_{x_l}^{x_r} \begin{pmatrix} x - x_0 \\ \\ y - y_0 \end{pmatrix} \times \begin{pmatrix} \frac{\partial I}{\partial y} \\ \\ -\frac{\partial I}{\partial x} \end{pmatrix} dxdy \\
&= \int_{y_t}^{y_b} \int_{x_l}^{x_r} \left(x - x_0\right)\left(-\frac{\partial I}{\partial x}\right) - \left(y - y_0\right)\frac{\partial I}{\partial y} dxdy \\
&= x_0 \int_{y_t}^{y_b} \int_{x_l}^{x_r} \frac{\partial I}{\partial x} dxdy + y_0 \int_{y_t}^{y_b} \int_{x_l}^{x_r} \frac{\partial I}{\partial y} dxdy \\
&- \int_{y_t}^{y_b} \int_{x_l}^{x_r} x \frac{\partial I}{\partial x} dxdy - \int_{y_t}^{y_b} \int_{x_l}^{x_r} y \frac{\partial I}{\partial y} dxdy \\
&= x_0 \int_{y_t}^{y_b} I\left(x_r, y\right) - I\left(x_l, y\right) dy + y_0 \int_{x_l}^{x_r} I\left(x, y_b\right) - I\left(x, y_t\right) dx \\
&- \int_{y_t}^{y_b} \left\{ \left[xI\left(x, y\right)\right]_{x_l}^{x_r} - \int_{x_l}^{x_r} I\left(x, y\right) dx \right\} dy \\
&- \int_{x_l}^{x_r} \left\{ \left[yI\left(x, y\right)\right]_{y_t}^{y_b} - \int_{y_t}^{y_b} I\left(x, y\right) dy \right\} dx \\
&= x_0 \int_{y_t}^{y_b} I\left(x_r, y\right) - I\left(x_l, y\right) dy + y_0 \int_{x_l}^{x_r} I\left(x, y_b\right) - I\left(x, y_t\right) dx \\
&+ 2 \int_{y_t}^{y_b} \int_{x_l}^{x_r} I\left(x, y\right) dxdy \\
&- \int_{y_t}^{y_b} x_r I\left(x_r, y\right) - x_l I\left(x_l, y\right) dy - \int_{x_l}^{x_r} y_b I\left(x, y_b\right) - y_t I\left(x, y_t\right) dx \\
&= 2 \int_{y_t}^{y_b} \int_{x_l}^{x_r} I\left(x, y\right) dxdy \\
&- \left(x_r - x_0\right) \int_{y_t}^{y_b} I\left(x_r, y\right) dy - \left(x_0 - x_l\right) \int_{y_t}^{y_b} I\left(x_l, y\right) dy \\
&- \left(y_b - y_0\right) \int_{x_l}^{x_r} I\left(x, y_b\right) dx - \left(y_0 - y_t\right) \int_{x_l}^{x_r} I\left(x, y_t\right) dx. \qquad (2.25)
\end{aligned}
$$

When $x_l = x_0 - w/2$, $y_t = y_0 - h/2$, $x_r = x_0 + w/2$ and $y_b = y_0 + h/2$, the eq. (2.25) can be written as follows:

$$\begin{aligned}\tau(x_0, y_0) = 2 &\int_{y_0-h/2}^{y_0+h/2} \int_{x_0-w/2}^{x_0+w/2} I(x, y)\, dxdy \\ &- \frac{w}{2} \int_{y_0-h/2}^{y_0+h/2} I(x_0 + w/2, y)\, dy \\ &- \frac{w}{2} \int_{y_0-h/2}^{y_0+h/2} I(x_0 - w/2, y)\, dy \\ &- \frac{h}{2} \int_{x_0-w/2}^{x_0+w/2} I(x, y_0 + h/2)\, dx \\ &- \frac{h}{2} \int_{x_0-w/2}^{x_0+w/2} I(x, y_0 - h/2)\, dx. \qquad (2.26)\end{aligned}$$

2.2.10 Similarity and Difference to Medial Axis

The medial axis of a shape, introduced by Blum [62], is defined as the set of all points where bi-tangent circle exists, which is equivalent to the set of all points where more than two closest points on the boundary of the shape exist. The definition of the medial axis is interpreted using the concept of time. Imagine that waves start from edges and spread uniformly with the same speed. These waves do not go through each other, *i.e.* waves are canceled when they collide. The contours that appear at the places where waves collide, are the medial axis. Figure 2.22 shows wave fronts and the medial axis for simple shapes.

As we can see from Fig. 2.11, the torque map also evolves over patch sizes as if waves spread from edges. Similarly to Fig. 2.22, Fig. 2.23 shows wave fronts of torque values for a line segment. A line segment lies horizontally at the center of image, and the wave front is computed as the set of locations for every vertical scan

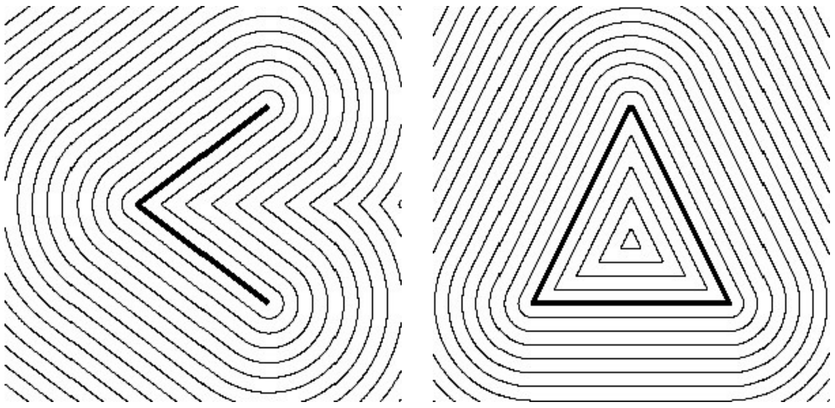


Figure 2.22: Wave Fronts and Medial Axis for Simple Shapes.

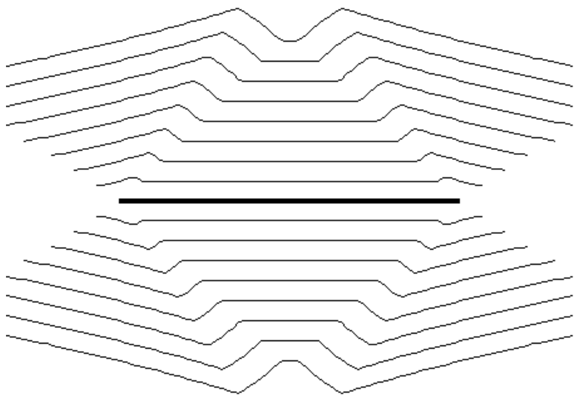


Figure 2.23: Wave Fronts of Torque for Line Segment.

where the torque is maximum in absolute value with several different disk patch radiuses. Due to the principle of superposition, torque values become high at the places where waves collide in the same phase. Therefore, we see lines in the torque value maps in Fig. 2.15 that look like medial axes. However, the wave front of torque over scales is not uniform due to effect of terminals of the line segment as can be seen in Fig. 2.23. Furthermore, waves are going through other waves according to the principle of superposition. A significant difference to the medial axis is that the torque measure captures the degree of surrounding edges.

Chapter 3

Application of Torque Operator

It is reasonable to expect, on the basis of current knowledge, that the visual processing necessary to find an object in a scene consists of three modules: visual attention, boundary detection, and foreground segmentation, as depicted in Fig. 3.1. Such an active approach is especially well suited for mobile robot applications. The attention mechanism is important for focusing the processing to the conspicuous region - the region of interest. In biological systems the attention is indicated by the fixation point. Segmenting foreground from background then requires to detect edges surrounding the fixation point. The torque mechanism can be used to emphasize contour edges and bias the segmentation towards these contours. In this section, we evaluate how much the proposed torque operator can improve each of these three steps mentioned above by comparing it against other methods in standard database settings.

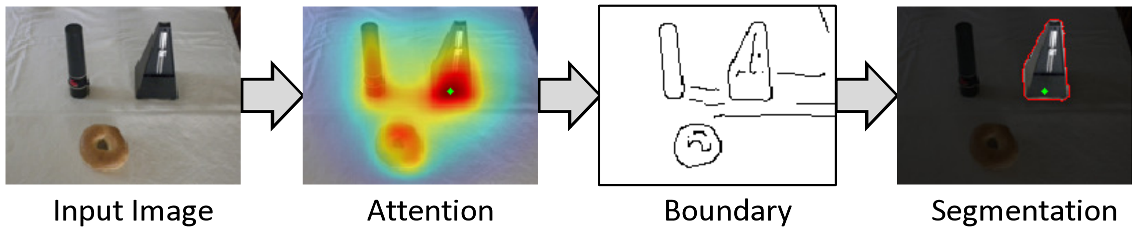


Figure 3.1: Visual Processing using Image Torque Operator

3.1 Efficient Torque Computation

The computation of torque is time consuming with a straightforward implementation. Next we introduce an efficient computational method for the torque using rectangular patches, using the concept of integral images. Let's say that the edge vector map is represented by maps of x and y components of edge vectors, $F^X(x,y)$ and $F^Y(x,y)$. The torque for an image patch can be written as follows:

$$\begin{aligned}\tau_P(x,y) &= \frac{1}{2|P|}\sum_{(u,v)\in P_{(x,y)}} (u-x)\,F^Y(u,v) - (v-y)\,F^X(u,v) \\ &= \frac{1}{2|P|}\left\{\sum_{(u,v)\in P_{(x,y)}} uF^Y(u,v) - \sum_{(u,v)\in P_{(x,y)}} vF^X(u,v)\right. \\ &\left. -x\sum_{(u,v)\in P_{(x,y)}} F^Y(u,v) + y\sum_{(u,v)\in P_{(x,y)}} F^X(u,v)\right\}. \end{aligned} \tag{3.1}$$

For a square patch, the summation is written as $\sum_{(u,v)\in P_{(x,y)}} = \sum_{v=y-w}^{y+w}\sum_{u=x-w}^{x+w}$. For computational efficiency, we can pre-compute the following arrays.

```
for $x = 0 \rightarrow$ (image width) $- 1$ do
    $H^X(x,0) \leftarrow F^X(x,0)$
    $H^Y(x,0) \leftarrow F^Y(x,0)$
    $G^X(x,0) \leftarrow 0$
    $G^Y(x,0) \leftarrow xF^Y(x,0)$
end for
for $y = 0 \rightarrow$ (image hight) $- 1$ do
    $H^X(0,y) \leftarrow F^X(0,y)$
    $H^Y(0,y) \leftarrow F^Y(0,y)$
```

```
        $G^X(0,y) \leftarrow yF^X(0,y)$
        $G^Y(0,y) \leftarrow 0$
    end for
    for $y = 1 \rightarrow$ (image hight) $- 1$ do
        for $x = 1 \rightarrow$ (image width) $- 1$ do
            $H^X(x,y) \leftarrow H^X(x-1,y) + H^X(x,y-1) + F^X(x,y)$
            $H^Y(x,y) \leftarrow H^Y(x-1,y) + H^Y(x,y-1) + F^Y(x,y)$
            $G^X(x,y) \leftarrow G^X(x-1,y) + G^X(x,y-1) + yF^X(x,y)$
            $G^Y(x,y) \leftarrow G^Y(x-1,y) + G^Y(x,y-1) + xF^Y(x,y)$
        end for
    end for
```

Then, the necessary summation in equation (3.1) within square patches can be computed simply as follows:

$$\sum_{u=x-w}^{x+w}\sum_{v=y-w}^{y+w} F^X(u,v) = H^X(x+w,y+w) + H^X(x-w-1,y-w-1) - H^X(x-w-1,y+w) - H^X(x+w,y-w-1). \tag{3.2}$$

$$\sum_{u=x-w}^{x+w}\sum_{v=y-w}^{y+w} F^Y(u,v) = H^Y(x+w,y+w) + H^Y(x-w-1,y-w-1) - H^Y(x-w-1,y+w) - H^Y(x+w,y-w-1). \tag{3.3}$$

$$\sum_{u=x-w}^{x+w}\sum_{v=y-w}^{y+w} vF^X(u,v) = G^X(x+w,y+w) + G^X(x-w-1,y-w-1) - G^X(x-w-1,y+w) - G^X(x+w,y-w-1). \tag{3.4}$$

$$\sum_{u=x-w}^{x+w}\sum_{v=y-w}^{y+w} uF^Y(u,v) = G^Y(x+w,y+w) + G^Y(x-w-1,y-w-1) - G^Y(x-w-1,y+w) - G^Y(x+w,y-w-1). \tag{3.5}$$

This way, the computational cost is significantly reduced because is not necessary to compute the summation for each position of the patch. The method work for arbitrarily rectangular patches by adapting the $\pm w$ above.

3.1.1 Approximation of Torque

Further efficiency in torque computation can be obtained by approximating the edge orientation to one of equally divided eight directions;

$$\theta_i = (i-1)\frac{2\pi}{8}, \tag{3.6}$$

where $i \in \{1,2,3,4,5,6,7,8\}$ is the index to the orientation. An edge vector of orientation θ_i at a pixel q is represented by an unit vector $e(q,i) = (\cos\theta_i, \sin\theta_i)^T$

For the data structure, the edge information can be sorted into two dimensional integer arrays of the same size as the image, where each element codes the index of the edge orientation if it corresponds to an edge pixel, otherwise it is set to zero. Furthermore, the displacement vector from the center of patch $p = (x_p, y_p)$ to an edge point $q = (x_q, y_q)$ is approximated as follows:

$$\begin{aligned} r_p(q,i) &= \left\{ (x_q - x_p)\cos\left(\theta_i - \frac{\pi}{2}\right) + (y_q - y_p)\sin\left(\theta_i - \frac{\pi}{2}\right) \right\} \begin{pmatrix} \cos\left(\theta_i - \frac{\pi}{2}\right) \\ \sin\left(\theta_i - \frac{\pi}{2}\right) \end{pmatrix} \\ &= \{(x_q - x_p)\sin\theta_i - (y_q - y_p)\cos\theta_i\} \begin{pmatrix} \sin\theta_i \\ -\cos\theta_i \end{pmatrix}. \end{aligned} \qquad (3.7)$$

This equation means that the displacement vector is approximated by its projection on the line perpendicular to the edge orientation. Referring to the definition of the torque value of an edge pixel in eq. (2.5), it is now approximated by the outer product of $e(q,i)$ and $r_p(q,i)$ as follows:

$$\tau_p(q,i) = \{(x_q - x_p)\sin\theta_i - (y_q - y_p)\cos\theta_i\} \cdot \delta(q,i), \qquad (3.8)$$

where $\delta(q,i) \in \{0,1\}$ is a binary indicator for the existence of the edge at a pixel q in the specific edge orientation indicated by i. Then, the torque for an image patch defined in eq. (2.6) is approximated as follows:

$$\tau_P = \frac{1}{2W_P^2} \sum_{q \in P} \sum_{i=1}^{8} \tau_p(q,i), \qquad (3.9)$$

where P is a set of pixels within the square patch. W_P is width (=height) of the patch P. Figure 3.2 explains these approximations. The bounding box in the figure represents the square patch. Two edge pixels a and b of orientation $\theta_4 = \frac{3\pi}{4}$ are depicted. It is indicated by $\delta(a,4) = 1$, $\delta(b,4) = 1$. The arm vectors from the center of the patch to the points a and b are approximated by their projections to the line in the orientation of $\theta_4 - \frac{\pi}{2}$, which are $r_p(a,4)$ and $r_p(b,4)$ respectively The torque value for a is positive, and it is negative for b in this case.

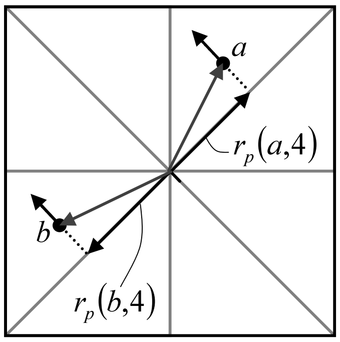


Figure 3.2: Efficient Approximation of the Torque. Orientation of edges at pixels a and b are approximated to $\frac{3\pi}{4}$ radians, and the displacement vectors to these points are approximated by their projections to the line in orientation of $\frac{\pi}{4}$ radians.

With this approximation, the torque for image patches can be efficiently computed. Assuming the orientation of edges are approximated and the edge maps are represented by a three-dimensional binary array $\delta(x,y,i)$, a three-dimensional array for the sum of edges is computed so that element (x,y,i) of this array holds the following value:

$$B(x,y,i) = \sum_{v=0}^{y} \sum_{u=0}^{x} \delta(u,v,i). \tag{3.10}$$

This computation can be done for each i in the following iterations:

```
$B_x(0,0,i) \leftarrow \delta(0,0,i)$
**for** $y = 0 \rightarrow$ (image height) $- 1$ **do**
```

```
        for $x = 1 \rightarrow$ (image width) $- 1$ do
            $B_x(x, y, i) \leftarrow B_x(x-1, y, i) + \delta(x, y, i)$
        end for
    end for
    for $x = 0 \rightarrow$ (image width) $- 1$ do
        for $y = 1 \rightarrow$ (image height) $- 1$ do
            $B(x, y, i) \leftarrow B(x, y-1, i) + B_x(x, y, i)$
        end for
    end for
```

Then, the sum of edges within a patch can be computed efficiently using B as follows:

$$\begin{aligned} B_P(x, y, i) &= \sum_{v=y-w}^{y+w} \sum_{u=x-w}^{x+w} b(u, v, i) \\ &= B(x+w, y+w, i) + B(x-w-1, y-w-1, i) \\ &\quad - B(x-w-1, y+w, i) - B(x+w, y-w-1, i), \end{aligned} \tag{3.11}$$

where $2w+1$ is the width of the patch. Another three-dimensional array is prepared, and the approximated torque values of edge pixels with respect to the origin of the image coordinate system is computed for each element of this array as follows:

$$\begin{aligned} m(x, y, i) &= \tau_o(x, y, i) \\ &= (x \sin\theta_i - y \cos\theta_i) \cdot \delta(x, y, i). \end{aligned} \tag{3.12}$$

Then, similarly to the sum of edges B, the sum of the torque value with respect to the origin of the coordinate system m is stored into an array M so that each element of the array holds the following value:

$$M\left(x, y, i\right) = \sum_{v=0}^{y} \sum_{u=0}^{x} m\left(u, v, i\right) \tag{3.13}$$

Then, the sum for patches can be computed similarly to B_P as follows:

$$\begin{aligned} M_P\left(x, y, i\right) &= \sum_{v=y-w}^{y+w} \sum_{u=x-w}^{x+w} m\left(u, v, i\right) \\ &= M\left(x+w, y+w, i\right) + M\left(x-w-1, y-w-1, i\right) \\ &- M\left(x-w-1, y+w, i\right) - M\left(x+w, y-w-1, i\right). \end{aligned} \tag{3.14}$$

With these arrays, the approximated torque for image patches in eq. (3.9) can be computed efficiently as follows:

$$\tau_P\left(x, y\right) = \frac{1}{2\left(2w+1\right)^2} \sum_{i=1}^{8} \left\{M_P(x, y, i) - \left(x \sin\theta_i - y \cos\theta_i\right) \cdot B_P\left(x, y, i\right)\right\}. \tag{3.15}$$

The equivalence to eq. (3.9) is confirmed as follows:

$$\begin{aligned} & M_P(x, y, i) - m\left(x, y, i\right) \cdot B_P\left(u, v, i\right) \\ &= \sum_{v=y-w}^{y+w} \sum_{u=x-w}^{x+w} m\left(u, v, i\right) - \left(x \sin\theta_i - y \cos\theta_i\right) \cdot \sum_{v=y-w}^{y+w} \sum_{u=x-w}^{x+w} b\left(u, v, i\right) \\ &= \sum_{v=y-w}^{y+w} \sum_{u=x-w}^{x+w} \left(u \sin\theta_i - v \cos\theta_i\right) \cdot b\left(u, v, i\right) - \left(x \sin\theta_i - y \cos\theta_i\right) \cdot b\left(u, v, i\right) \\ &= \sum_{v=y-w}^{y+w} \sum_{u=x-w}^{x+w} \left\{\left(u-x\right) \sin\theta_i - \left(v-y\right) \cos\theta_i\right\} \cdot b\left(u, v, i\right) \\ &= \sum_{q \in P} \tau_p\left(q, i\right), \end{aligned} \tag{3.16}$$

where $q = (u, v)$, $p = (x, y)$ and $P = \{(u, v) \,| x - w \le u \le x + w, y - w \le v \le y + w\}$. The computational cost is independent from the patch size, and computing the torque for all patches in the image is linear in the number of pixels $O\left(n\right)$.

3.2 Pre-processing for Application

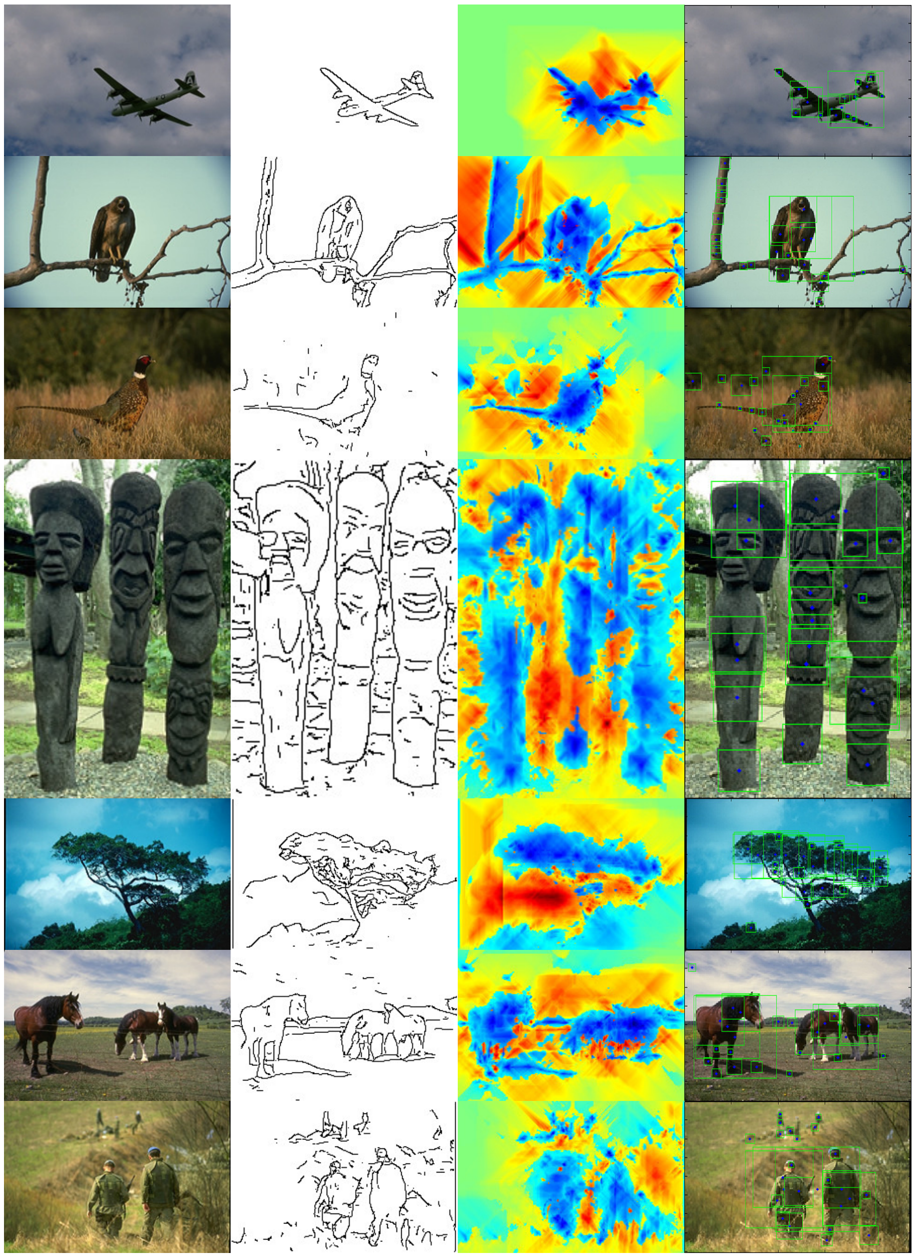
Before discussing applications to vision problems, we start with a discussion on several potential pre-processing operations using the torque operator. These pre-processing operations are based on the properties introduced in Chapter 2, and could be utilized for enhancing important structured surrounding edges. The processed edge maps would be favorable for most applications, where it is necessary to identify the object location in the image. Starting with torque extrema in actual images, ideas of strengthening and linking of edges, and rough image segmentation by connected component using the torque are introduced with examples in this section.

3.2.1 Torque Extrema for Images

As we discussed in Section 2.2.4, a torque extrema tends to indicate the existence of interesting structured edges, which are expected to be part of object boundaries. In this section, we discuss again this property of the torque operator using examples of actual images instead of synthesized shapes. In this experiment, edges are detected by the method proposed by Martin *et al.* [63]. The torque value is computed at every point over multiple sizes of image patches, and local extrema are found in the torque volume. We used test images from the Berkeley dataset [63]. The patches used for torque computation were square, and the side length varied from 3 to 91 pixels, while the images were shrunk to 161×241, which is half the size of the original image in the dataset. The largest image patch used, covers about 21% of the entire image. Figure 3.3 shows test images, edge maps, torque value maps, and the locations of extrema and corresponding patch size. As can be seen in (c), negative torque regions match the object regions for these test images, though it depends on the relative brightness of the objects to the background, because the sign of the torque depends on the orientation of edges. An interesting property of the torque operator is that extrema are not located just at the dense edges, but at places surrounded by edges. For example, in the image of statues, some parts inside the statues are less textured, and no edges are extracted as seen at the bottom parts of the statues. However, the local minima in the torque volume spots the location of structured surrounding boundaries at the lower part of the statues. The second interesting property of the torque operator is that these extrema indicate sizes of structured edges. For these example test images, local minima and corresponding patch sizes are indicating locations and sizes of part of object delineated by the object boundaries in the edge maps. It can be seen that these patches associated with local minima cover most of the object regions. The locations and sizes of objects are not perfectly found using only by the torque operator, however these properties of indicating roughly location and size of structured edges are favorable for further image processing, such as visual attention and object segmentation.

3.2.2 Edge Strengthening

Once we find the torque extrema, we expect that those extrema are produced by surrounding structured edges. Therefore, it is expected that edges producing those extrema are likely more important boundary edges than other edges. For each



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

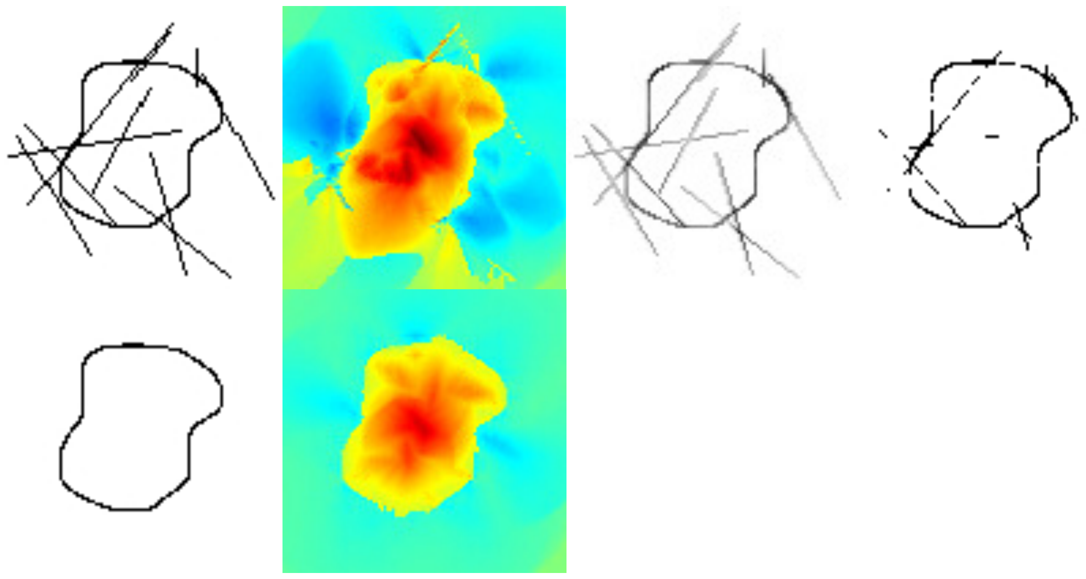
Figure 3.3: Torque Value Maps and Extrema in Torque: (a) Original test images. (b) Edge maps detected by the method proposed by Martin *et al*. [63]. (c) Torque value maps. Red color represents positive, and blue color represents negative. (d) Local minima in torque volume. The dots show the location of extrema in image space, and corresponding square shows the size of the patch producing the extrema.

extremum, we can specify the center of the image patch and the size of the patch. Then, with the given patch location and size, we think of taking the torque computation the other way around. While all the effect of oriented edges as rotational force to the center of the patch are accumulated to compute the torque for an image patch, now the rotational force at each edge pixel to each extremum is computed and accumulated over extrema at each edge pixel. The torque value is computed for each edge pixel within the patch associated with an extremum, and normalized by the factor of the patch size. In other words, the torque extremum for the image patch is decomposed into component corresponding to the different edge pixels. We call the decomposed torque value the 'contribution to the torque extremum'. The contributions to all extrema are summed up at every edge pixel. The contribution quantity at an edge pixel is computed as follows:

$$\upsilon\left(q\right)=\sum_{\{p|q\in P(p,S(p))\cap p\in V\}}\frac{\tau_{pq}}{2\left|P\left(p,S\left(p\right)\right)\right|}, \tag{3.17}$$

where $\upsilon\left(q\right)$ is the accumulated contribution value to torque extrema at a point q. $P\left(p,S\left(p\right)\right)$ is the set of pixels inside the image patch specified by the center of the patch p and the scale of the patch $S\left(p\right)$, with which the torque extrema is produced at p. V is the set of torque extrema positions. Edges with higher contribution to the extrema are considered parts of structured edges or object boundaries, but edges with lower contribution to the extrema can be regarded as less-important edges. Therefore, the given edges can be strengthened and weakened based on the contributions to the extrema. We first show this idea for a simple synthesized edge map, and experiments with actual images will be shown later. Figure 3.4 shows the synthesized example. Figures for an edge map consisting of a single closed boundary edge and some random edges are shown in the upper row, and figures for an edge map containing only a single closed boundary edge are shown in the lower row for reference. The edge maps are shown in (a). The corresponding torque value maps are shown in (b). The edge map after the edge strengthening process is shown in (c) only for the first row. In order to better distinguish edges strengthened in this process from other edges, the strengthened edge map is binarized by a threshold and shown in (d) for the first row. The binarization is only for the sake of better understanding of the effect of this process. We see that mostly important closed edges remains after the binarization because these edges were strengthened and other edges were weakened.

A surface of the foreground object, uniform in appearance, is generally either brighter or darker than the background. However, this is often not true for the whole object, and it is possible that some parts of the object surface are brighter and other parts are darker than the background. However, without top-down knowledge in composition of object surfaces, the assumption that roughly objects are either brighter or darker than the background is reasonable for the purpose of finding interesting edges at a first step of image processing. With this assumption, it is beneficial to separate the contributions of edge pixels to torque extrema into to torque maxima and to torque minima. It is because boundaries of brighter object produce maxima in torque, and vice versa. For the case of Fig. 3.4, they are not separated because it was edge-base analysis. For each pixel, accumulated contributions to torque maxima



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

Figure 3.4: Example of Edge Strengthening for Synthesized Edge Map: (a) Synthesized edge map that consists of a single closed boundary and some random edges. (b) Torque value map for the edge map in (a). (c) Strengthened edge map directly corresponds to contribution to torque extrema at each edge point. (d) Strengthened edge map is binarized by thresholding to emphasize visually that the closed boundary are mainly strengthened, and the random edges are weakened by the process.

and torque minima are respectively computed as follows:

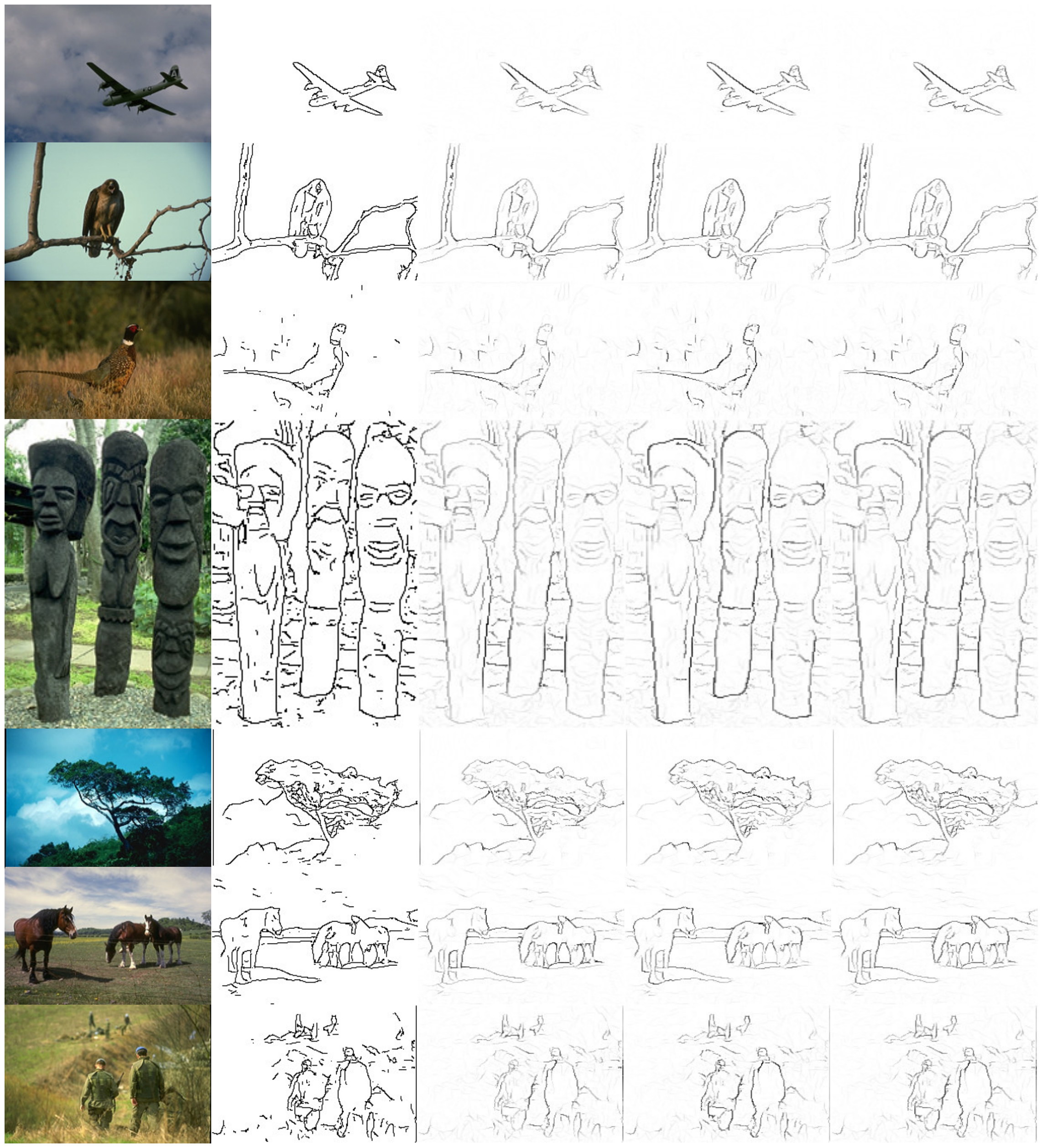
$$\upsilon^{+}(q) = \sum_{\{p|q \in P(p,S(p)) \cap p \in V^{+}\}} \frac{\tau_{pq}}{2\left|P\left(p, S\left(p\right)\right)\right|}, \tag{3.18}$$

$$\upsilon^{-}(q) = \sum_{\{p|q \in P(p,S(p)) \cap p \in V^{-}\}} \frac{\tau_{pq}}{2\left|P\left(p, S\left(p\right)\right)\right|}, \tag{3.19}$$

where $\upsilon^{+}(q)$ is the accumulated contribution to torque maxima at q, and $\upsilon^{-}(q)$ is the accumulated contribution to torque minima at q. $P(p, S(p))$ is the set of pixels inside the image patch specified by the center of the patch p and the scale of patch $S(p)$ with which the torque extrema is produced. V^{+} and V^{-} are set of pixels accompanies with positive and negative torque extrema respectively. This computation can be done as follows; Visiting each either maxima or minima of torque, a patch is defined by the visited extrema with the corresponding center of patch p and the scale of $S(p)$. Then, the torque value τ_{pq} for each edge pixel q within the patch P is computed. As visiting every maxima or minima, the computed torque values at the same edge pixel are summed up.

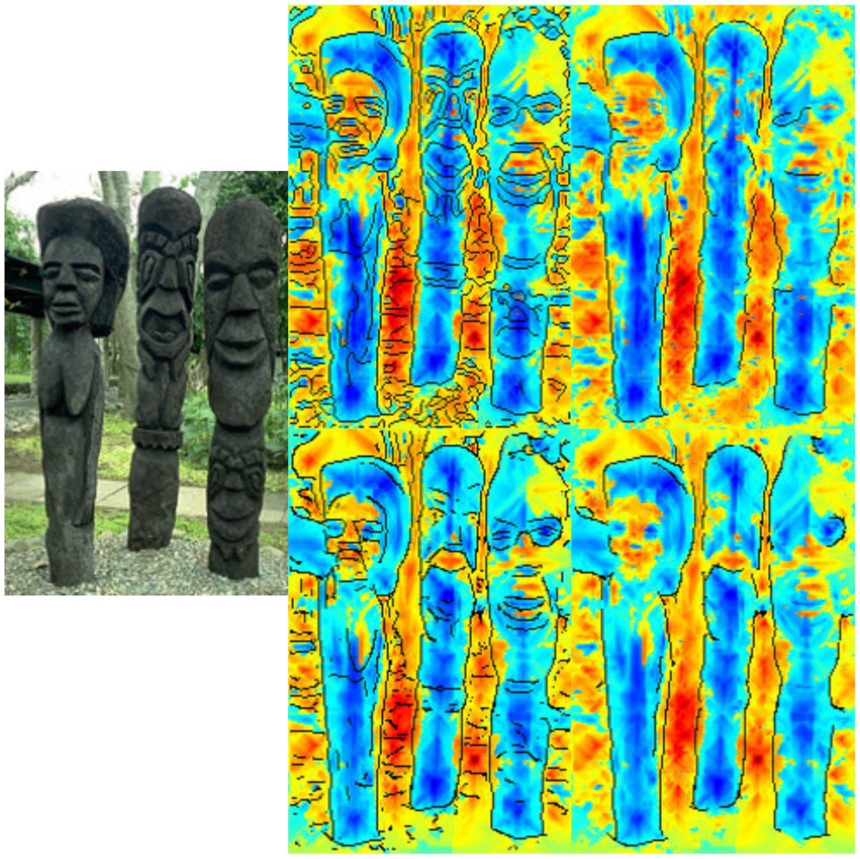
In case we could assume that a object is either brighter or darker in image. One of two contribution quantities to torque extrema, either maxima or minima is more efficiently utilized for strengthening important edges. Figure 3.5 shows examples of the idea of splinting the edge strengthening by extrema into edge strengthening by torque maxima and edge strengthening by torque minima. The test images are shown in (a). The detected edge maps before the edge strengthening process are shown in (b). The strengthened edge maps using torque maxima are shown in (c). The strengthened edge maps using torque minima are shown in (d). For these test images, objects of interest are relatively darker than background. Therefore, object boundaries tends to be more well emphasized by edge strengthening using torque minima than the other one using torque maxima. It also can be seen that unstructured edges, such as small fragments of edges, are weakened by this process. The average edge maps between these two strengthened edge maps using torque maxima and minima are shown in (e) for reference. In this experiment, the images of strengthened edge maps are generated by simply normalizing the contribution quantities based on maxima and minima respectively between zero and one.

By the edge strengthening, important edges can be emphasized, but complete boundaries are still difficult to be detected. For the possibility of further improvement, we see more on the relationship between edges and torque value map. The edges are overlaid onto torque value map in Fig. 3.6. Since torque value is color coded so that warm colors are positive and cold colors are negative, the boundary between positive and negative can be seen by the color change in the figure. In the column (b) of the figure, the upper image shows Canny edges overlaid onto the torque value map. The lower image shows pb edges by Martin *et al.* [15] overlaid on the torque value map. Similarly, the strengthened edges based on Canny edges are shown on the value map in upper image in column (c). The strengthened edges based on pb edges edges are shown on the value map in lower image in column (c). The first impression from this visualization is that torque values inside the object tends to negative, and outside object are in positive. Therefore, connected components of respective positive and negative regions could be clue for segmentation and improvement of strengthened edges for complete boundaries.



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

Figure 3.5: Strengthened Edge Maps Example: (a) test images. (b) detected edge maps. (c) strengthened edge maps using torque maxima. (d) strengthened edge map using torque minima. (e) average edge maps of (c) and (d).



(a) (b) (c)

Figure 3.6: Edges and Torque Value Map: (a) Original images. (b) upper: Canny edges overlaid on the torque value map, lower: pb edges by Martin *et al*. [15] overlaid on the torque value map. Warm color is positive, and cool color is negative in torque value, and the black lines are edges. (c) upper: Cleaned strengthened edges based on Canny edges overlaid on the torque value map, lower: Cleaned strengthened edges based on Berkeley edges overlaid on the torque value map.

3.2.3 Connected Components

In the previous section, potential usefulness of connected component is mentioned. We will further look deeper how connected components look like. At the beginning it is explained how connected components are computed given a torque value map though it is straightforward. Figure 3.7 shows the steps to compute the connected components. First, the torque map is split into positive region and negative region. In the figure, the black and white images represent the positive and negative torque regions. These regions are simply identified by torque $\tau > 0$, and $\tau < 0$, respectively. Therefore, they are complementary each other except points where $\tau = 0$. Pixels are labeled '1' if $\tau > 0$ and '0' otherwise for positive torque regions, and labeled '1' if $\tau < 0$ and '0' otherwise for negative torque regions. Second, connected components of positive torque regions, and connected components of negative torque regions are computed respectively. A connected component, here, is a set of pixels labeled '1' and connected to another pixel labeled '1' in four neighbors. In the figure, connected components are visualized using different colors where pixels in the same color belong to the same connected component. Third, these connected components for positive and negative torque regions are combined into one image because they are not overlapped. This image can be seen as a rough object segmentation of original image.

For more visual inspection on connected components, torque value maps are computed for seven images from Berkeley image dataset [63], and connected components of positive and negative torque value regions are generated to see possibility

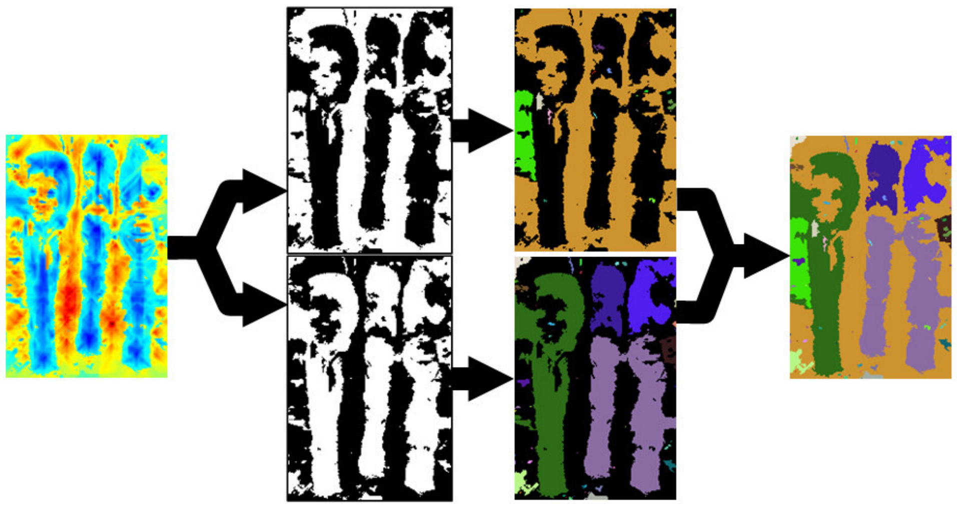


Figure 3.7: Process of Computing Connected Components: The torque value map is split into positive and negative torque value region first, and then they are segmented into connected components. Finally, these connected components are combined into segmentation in one image.

of rough object segmentation and edge linking. The test images and the connected components are shown in Fig. 3.8. In this experiment, the test images in the dataset was resized into half in width and hight, and torque were computed using square patches. The width of square patch varies odd number between 3 and 45 pixels. Since torque values inside a closed boundary tends to be the same sign, it is expected that connected components of positive or negative torque region can be a clue to find object regions and complete boundaries.

As it is seen in Fig. 3.8, the connected components has potential to be used as the first step of visual processes, such as object detection, boundary detection, and segmentation. At the computation of connected components of torque value, the boundaries of positive and negative torque regions carry important information on segmentation. In order to look deeper on the boundaries of positive and negative

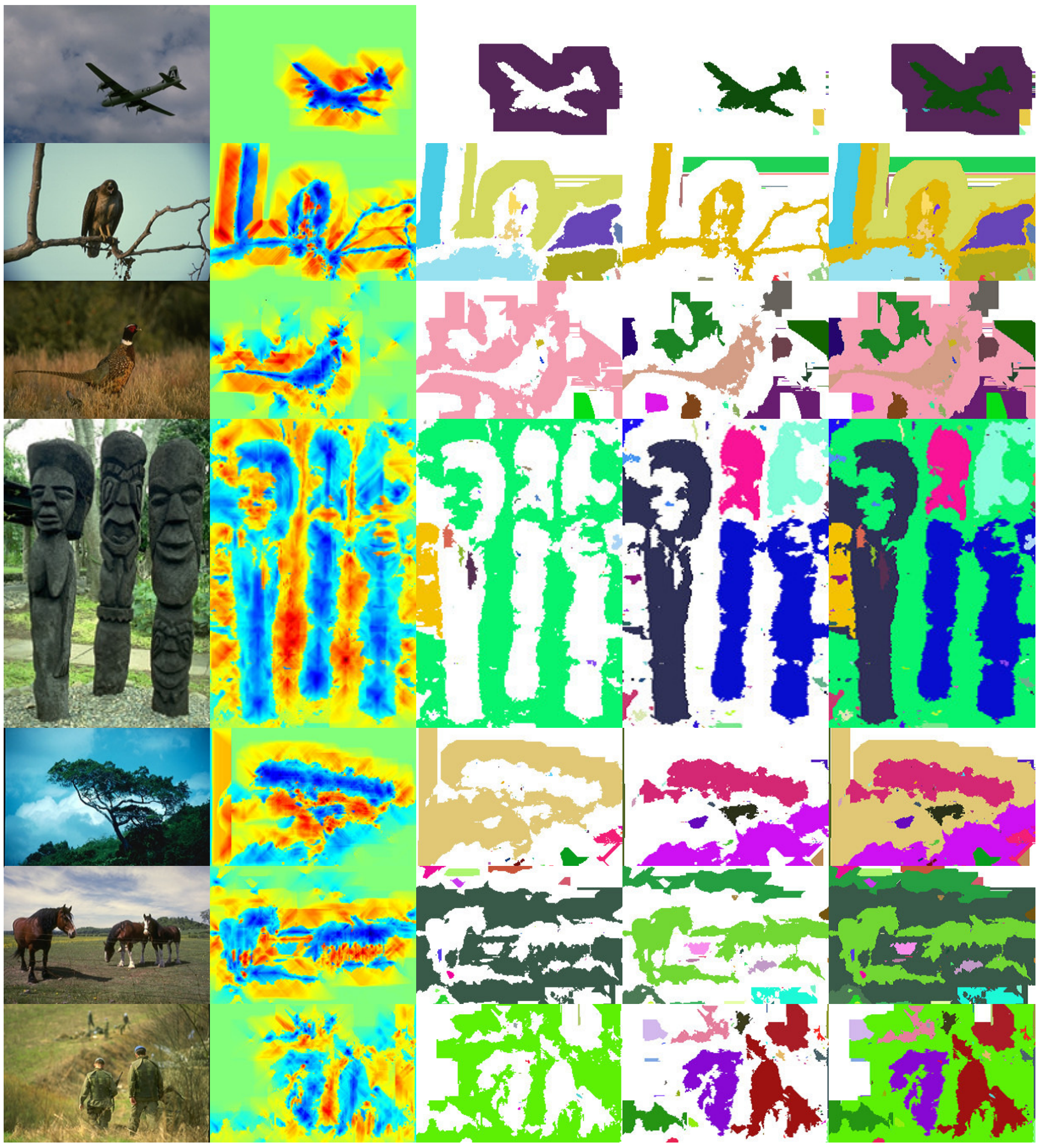


Figure 3.8: Connected Components in Torque Value Map: (a) Test images. (b) Torque value maps, where red color assigned to positive and blue color assigned to negative torque. (c) Connected components of positive torque regions. (d) Connected components of negative torque regions. (e) Connected components of both positive and negative torque regions, which are generated by combining (c) and (d). These images can be seen as rough object segmentation based on torque value. For the images of connected components, pixels in the same color belong to the same component.

torque regions, we compute regions where torque values are close to zero and shown in Fig. 3.9. Regions where torque value are close to zero are colored in black in the figure. Nearly zero here means that the absolute torque value is less than ϵ in torque value map, and ϵ varies as an parameter. According to Fig. 3.9, the zero torque regions, the black area in the figure, with ϵ between 0.15 and 0.25 look better to delineate objects than other range of ϵ value.

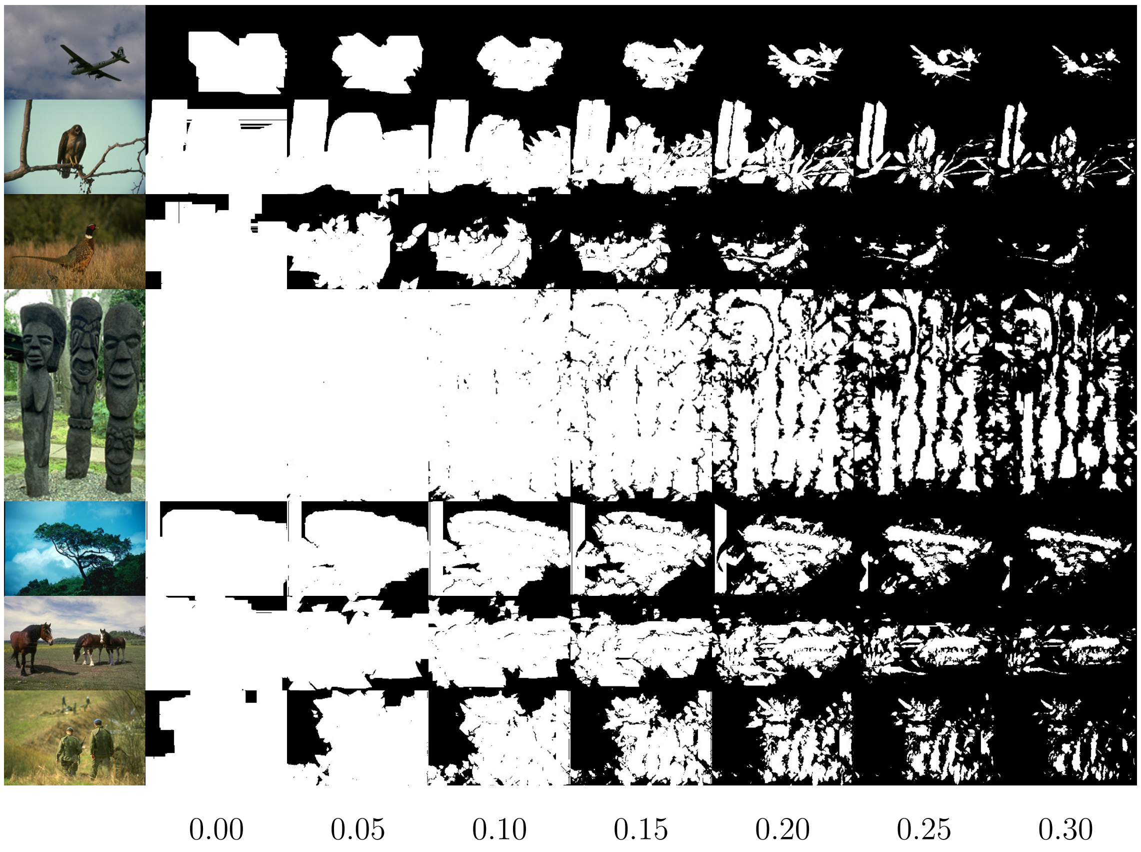


Figure 3.9: Zero Torque Region: Left most column shows test images. The other columns show regions where value in torque value map is between $-\epsilon$ and ϵ in black. The value for ϵ is shown at bottom of each column.

The connected components were generated based on the positive $\tau > 0$ and negative $\tau < 0$ torque regions previously. One of problems of this way is multiple regions are linked by narrow bridges, and became one connected component. These bridges are often caused by pixels with torque values relatively close to zero. As we see in Fig. 3.9, the zero torque regions with small margin between $-\epsilon$ and ϵ are expected to cut these narrow bridges. Then, positive and negative regions are now computed by $\tau > \epsilon$ and $\tau < -\epsilon$ respectively instead of zero for the ϵ. Connected components computed in this way are shown in Fig. 3.10. It could be solution to split connected component with narrow bridges into reasonable mass of regions. In some sense, connected components shows potential to segment an image to parts of objects, but further study is required to utilize it for segmentation and detecting boundaries. Next section will show possibility of indirect usage of connected components as initial state of successive image processing.

3.2.4 Edge Linking

In the previous section, we showed the potential usefulness of connected components to obtain a rough sketch of objects in an image. In this section, we introduce a method utilizing the connected component of torque value for edge linking.

The goal of linking edges is finding appropriate closed boundaries. In this sense, this problem of linking edges can be interpreted to finding closed loops which coincide with most of edges. Level set algorithm is one of methods that could be used for this purpose. This method needs a good initial region of objects, usually given as a bounding box. Since torque value map indicate rough location of objects,

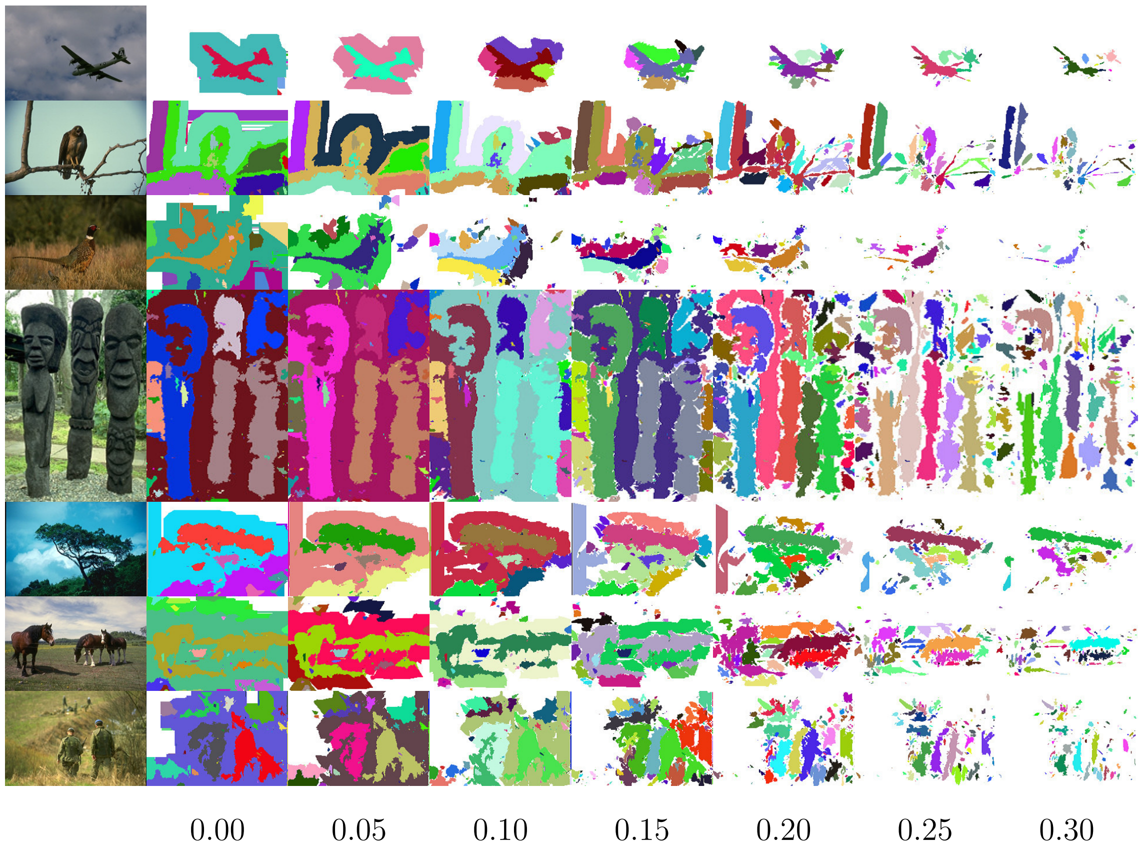


Figure 3.10: Connected Components with Zero Torque Regions: Connected components are computed on positive and negative torque region defined by $\tau > \epsilon$ and $\tau < -\epsilon$ respectively. The number below each column correspond to ϵ. Pixels in same color belong to the same connected component. Regions in black do not belong to neither positive nor negative regions. Canny edges are used for computing torque for the upper row, and Berkeley edges are used for lower row.

this information can be used as initial region to find closed contours. We can use the connected components of torque value for this purpose, and then it is expected that boundaries starting with the connected components are adjusted to fit to edges appropriately by the level set method. In this experiment, Chan-Vese method [64] is used for finding boundaries by level set. In order to perform the Chan-Vese method, gray scale images are used, but not edge maps directly. However, the edges are used in torque computation and initial segmentation, *i.e.* edges are used indirectly in the level set method.

The test image used in this experiment is shown in Fig. 3.11 (a). First, we tried to use negative torque regions as initial segmentation, which is shown in (b). The regions in white are the initial segmentation. The segmentation obtained by Chan-Vese method using the negative torque regions (b) is shown in (c). In the figure, strengthened edge map introduced in Sec. 3.2.2 are overlaid in blue. It can be seen that the edges do not form complete closed boundaries. Since edges are indirectly used in the level-set method, most of boundaries of the level-set segmentation are coincide with the strengthened edges. Second, we tried to use a connected component as initial segmentation. The connected component corresponding to the middle statue was manually selected, which is shown in (d). The segmentation using this connected component is shown in (e). This way, the boundary of the middle statue mostly coincide with the strengthened edges were found. Third, for a reference, an manual initial segmentation shown in (f) is generated. Then, the segmentation using (f) is shown in (g).

The segmentation shown in Fig. 3.11 (c) is not perfect, but reasonably good in

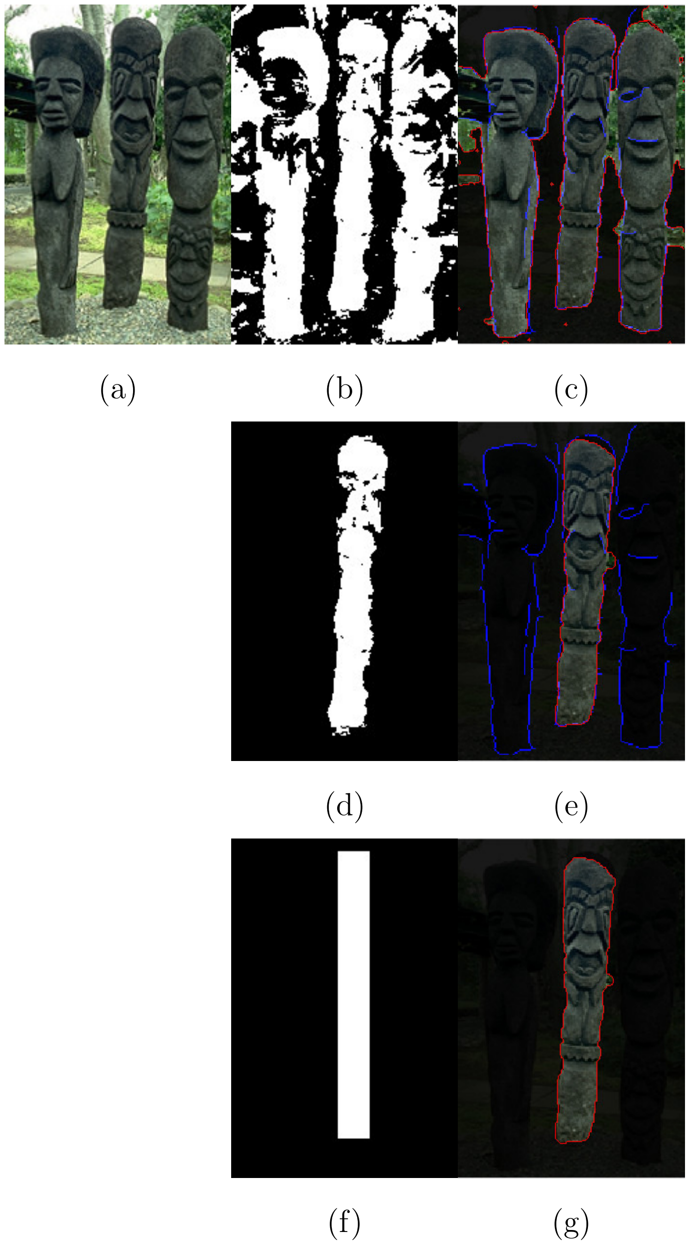


Figure 3.11: Segmentation by Chan-Vese Method [64] using Torque Value: (a) Original images. (b) Negative torque regions (c) Segmentation using all negative torque region shown in (b) as initial segmentation. (d) A manually selected connected component in negative torque region. (e) Segmentation using a connected component shown in (d) as initial segmentation. (f) Manually set box region for segmentation of the middle statue. (g) Segmentation using manually given box shown in (e) as initial segmentation. Red lines are boundaries of segmentation, and blue lines are the strengthened edges by torque.

the sense of using only low level cues. As seen in Fig. 3.11 (g), if a good initial guess of object location is given, a good segmentation can be obtained for this image. In other words, one of benefits in torque measure for segmentation and boundary detection can be addressed as providing initial segmentation. As shown in results, it is worth to note that obtained segmentation boundaries mostly coincide with the strengthened edges.

3.3 Visual Attention

The torque measure tends to have high values in magnitude at points surrounded by boundary edges with patches of the structured edges. This property of the torque measure is expected to be useful as a cue for bottom-up visual attention. Saliency maps for visual attention on test images of objects on table are computed using torque measure. We used the following non-linear transformation from torque value map to saliency map.

$$s = \begin{cases} \ln\left(-c \cdot \tau + 1\right) & \text{if } \tau < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.20}$$

where τ is torque value and s is saliency value. c is a positive constant. In this experiment, an assumption that background is brighter than foreground objects is made in the transformation. Therefore, the saliency value is high as $-\tau$ is large. Furthermore, the torque-based saliency maps are visually compared with saliency maps by a traditional method by Itti *et al.* [20] and a graph-based visual saliency (GBVS) by Harel *et al.* [21]. The Fig. 3.12 shows saliency maps computed by these three methods overlaying onto test images. Higher saliency value is coded in red color in the figure.

The traditional method by Itti *et al.* often shows higher saliency value at location close to edges because changes of color and intensity occur at edges. Graph-based method by Harel *et al.* tends to show saliency region only around center of the images and higher value at around edges as well. Torque-based method often shows object-wise saliency region, but not broad region like other methods. Some salient regions by torque method show salient regions that do not belong to objects, but some dark region compare to surrounding region such as shadow on table or floor. Assumption that objects are darker than background in images are used for torque method this time, but probably this could be fixed simply checking dominant sign of torque, or sign of torque around boundary of image and center of image.

In the previous experiment, ideas of extrema and scale for torque-based method are not taken into account. Therefore, extrema and their scale are visualized as attention regions in this experiment. Signs of torque values are flipped under an assumption that foreground object is darker than background in image. Only local maxima larger than threshold are selected as locations of attention, and visualized with scale associate with the local maxima in Figure 3.13. The figure shows that most of object regions are covered by the attention regions. It indicates that torque measure is good to represent region of interest. Some of regions wrongly found at image boundary are due to boundary effect of torque computation and will be easily fixed.

In the following experiment, we computed two torque-based saliency maps

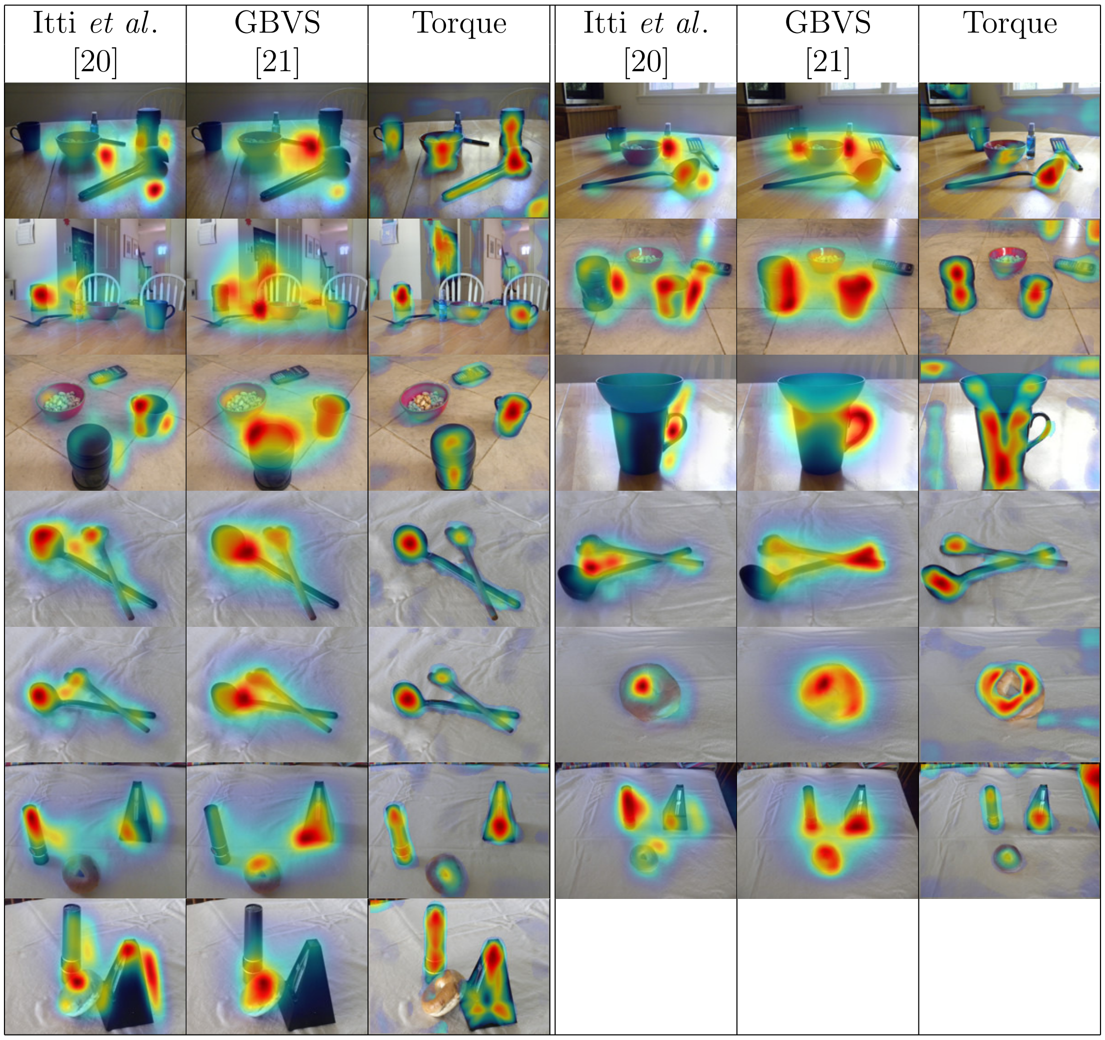


Figure 3.12: Visual Attention Comparison: Saliency maps computed by three different methods are visualized by overlaying onto each test image respectively. Here, traditional method is the one proposed by Itti *et al.* which compute saliency based on center-surround difference in feature maps. The graph-based method here is proposed by Harel *et al.* and dissimilarity of features in regions are computed using a graph-based approach to identify conspicuous regions.

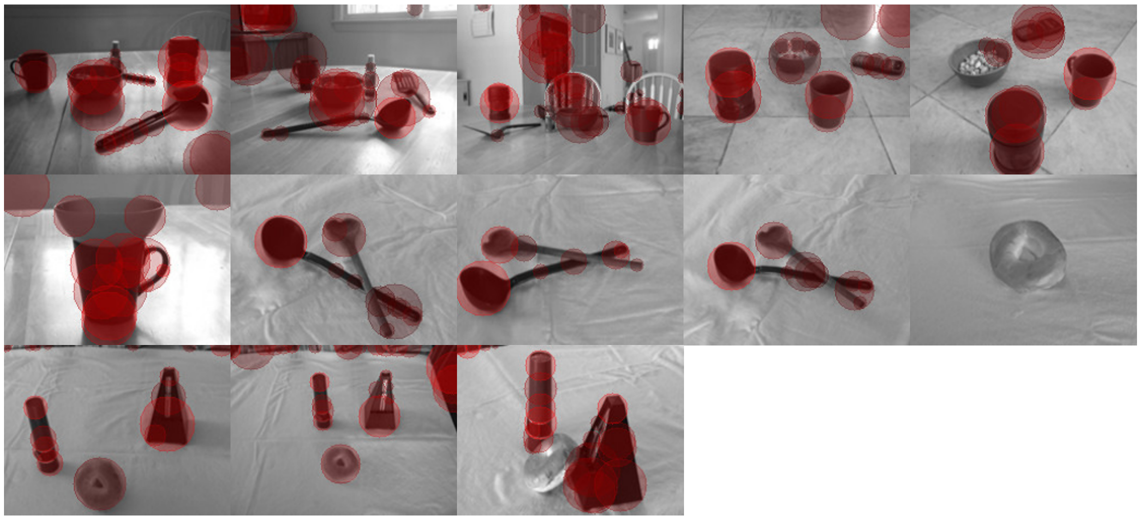


Figure 3.13: Visual Attention by Torque Extrema: test images are overlaid with discs representing location of extrema in torque volume and associating size. Brighter red color inside disc represents higher absolute torque value.

for each test image; one is generated by mixture of Gaussian where each Gaussian distribution is centered at each extremum value in the torque volume, and the other is weighted sum of the generated saliency map and the GBVS.

$$s^{GBVS+Torque} = \alpha s^{Torque} + (1-\alpha)\, s^{GBVS}, \tag{3.21}$$

where s^{GBVS}, s^{Torque}, and $s^{GBVS+Torque}$ are saliency maps by GBVS, generated from torque value map, and their weighted sum respectively. We used 0.3 for the weight parameter α so that it weights 0.3 for the torque-based saliency map and 0.7 for the GBVS. The computed torque-based saliency maps are normalized in $[0, 1]$.

The torque-based saliency maps are quantitatively compared with saliency maps of Itti *et al*. and the GBVS. We used eye tracking data by Judd *et al*. [65] to generate ground truth saliency maps. Fixation points were extracted from the data, and saliency maps were generated as mixture of Gaussian distributions where each Gaussian distribution is centered at each fixation point. The generated saliency maps are normalized in $[0, 1]$.

The ground truth saliency maps are binarized by a threshold (=0.5) for the following quantitative measure. We used a set of thresholds equally distant in $[0, 1]$, and the computed saliency maps are binarized by each threshold to measure accuracy. We measured precision and recall of the computed saliency map as follows:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{FP + FN}, \tag{3.22}$$

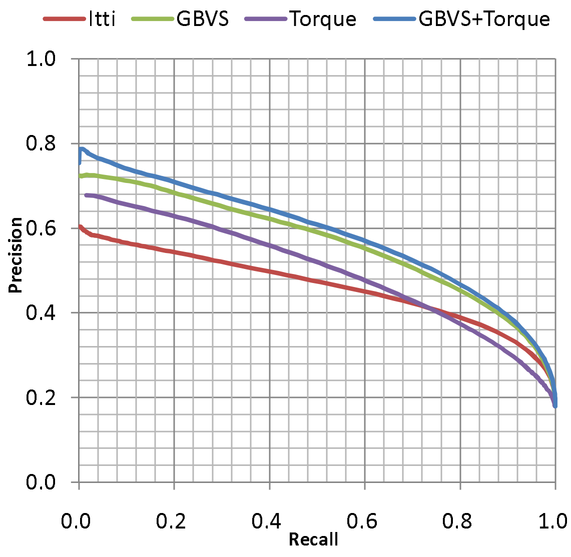
$$TP = \left|S \cap \mathcal{G}\right|, FP = \left|S \cap \overline{\mathcal{G}}\right|, FN = \left|\overline{S} \cap \mathcal{G}\right|, \tag{3.23}$$

where S is the computed saliency map binarized by a threshold. $\mathcal{G}$ is the binarized ground truth saliency map. P and R denote precision and recall respectively. TP, FP, FN are true positive, false positive, and false negative respectively.

We resized the test images so that the length of the shorter side of the images is 150 pixels, and the standard deviation of the Gaussian distribution applied to generate ground truth and torque-based saliency maps were both set to 25 pixels.

ROC curves and maximum F-measures of them over valid 898 test images in the dataset are compared and shown in the Fig. 3.14. In the figure, labels of Itti, GBVS, Torque, and GBVS+Torque indicates the model of Itti *et al.*, graph-based visual saliency by Harel *et al.*, and torque-based method, and combination of GBVS and torque, respectively. Examples of computed saliency maps are shown with ground-truth in Fig. 3.15.

The quantitative comparison shows that the torque-based attention by itself does not outperform GBVS. This is not surprising because GBVS is a sophisticated method that includes a series of features such as texture, and it codes relations between different parts of the image. Attention is related to recognition, and texture is known to play a very important role in recognition. It has been shown by current computer vision applications to be more powerful than contour. However, the torque measure as an additional mid-level visual cue improves the quality of GBVS.



(a)

method	F-measure
Itti	0.53
GBVS	0.59
Torque	0.54
GBVS+Torque	**0.60**

(b)

Figure 3.14: Evaluation of Attention Models. (a) Precision-recall curves. (b) Average of maximum F-measure.

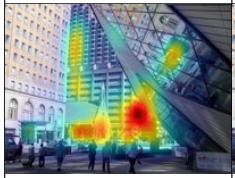
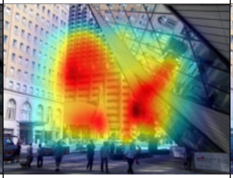
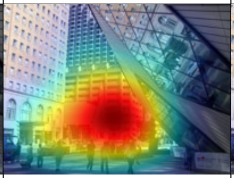
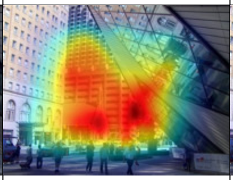
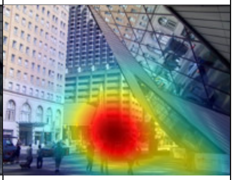
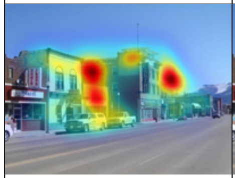
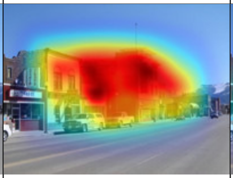
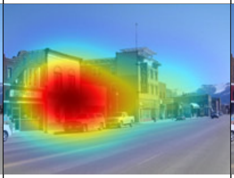
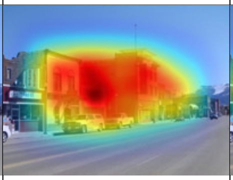
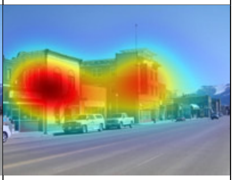
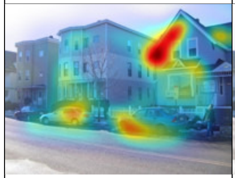
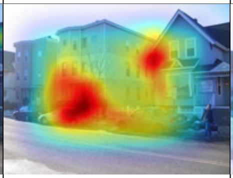
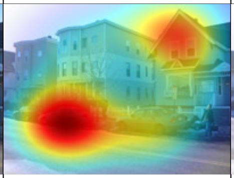
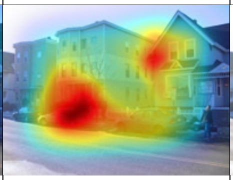
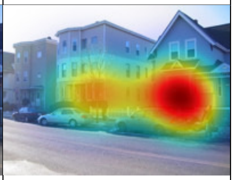
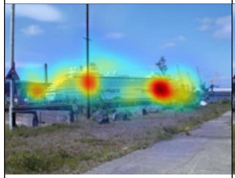
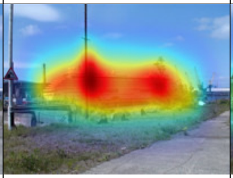
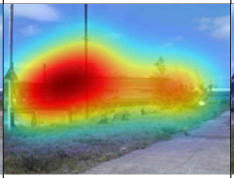
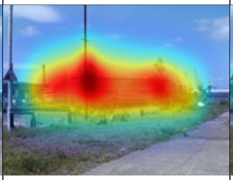
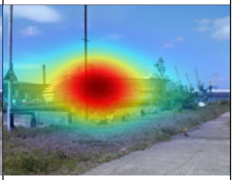
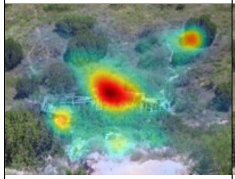
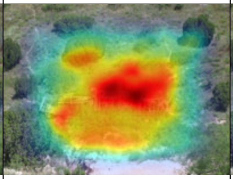
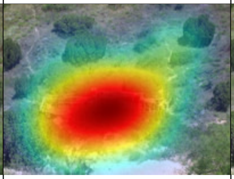
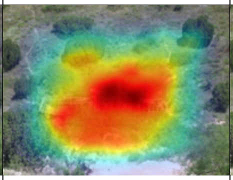
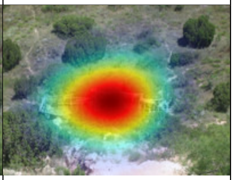
Itti *et al.*	GBVS	Torque	GBVS+Torque	Ground truth
				
0.608	0.486	**0.687**	0.544	
				
0.700	0.718	0.789	**0.794**	
				
0.558	**0.604**	0.481	0.585	
				
0.775	0.795	0.774	**0.811**	
				
0.690	0.760	0.815	**0.818**	

Figure 3.15: Examples of Visual Attention. Saliency maps computed by four different methods and ground-truth saliency map are visualized by overlaying onto test image respectively. The maximum F-measure is shown for each saliency maps.

3.4 Boundary Detection

Torques for image patches are computed using edges. The contribution of the edge point to multiple torque values of different patches can be written as follows:

$$\upsilon\left(q\right)=\sum_{\{p|q\in P(p,S(p))\cap p\in V\}}\frac{\tau_{pq}}{2\left|P\left(p,S\left(p\right)\right)\right|}, \tag{3.24}$$

where $\upsilon\left(q\right)$ is the accumulated contribution value to torque extrema at a point q. $P\left(p,S\left(p\right)\right)$ is set of pixels inside the image patch specified by the center of the patch p and the scale of the patch $S\left(p\right)$, with which the torque extrema is produced at p. V is the set of torque extrema positions. Extrema of the torque value indicate the existence of surrounding edges around the center of the patch at the corresponding scale. Therefore, by computing each edge point's contribution in eq. (3.24) to extrema, the quantities at edges of surrounding structure are expected to be high. Using these contribution quantities to torque extrema, the original edges can be strengthen or weaken to emphasize important edges. Assuming objects have surrounding edges of boundaries, these strengthened edges are expected to represent object boundaries. Examples of strengthened edges are shown in Fig. 3.16. The Canny edges shown in (b) are used to compute torque in this experiment. The strengthened edges tend to be stronger at boundary edges of objects, while weaker at texture edges. The computed edge's contribution to torque is normalized into $[0,1]$. Edges are strengthened by combining the original edges with the contribution value as follows:

$$d_s=\frac{1}{1+e^{-(c_0+c_1d_o+c_2d_t)}}, \tag{3.25}$$

where d_o is the original edge intensity, d_t is the normalized torque contribution. c_0, c_1, and c_2 are constants. We used machine learning method using 200 training images in Berkeley dataset [63] to determine the parameters, $c_0, c_1, and c_2$. In this experiment, -2.54, 1.86, and 2.69 were used for these parameters respectively.

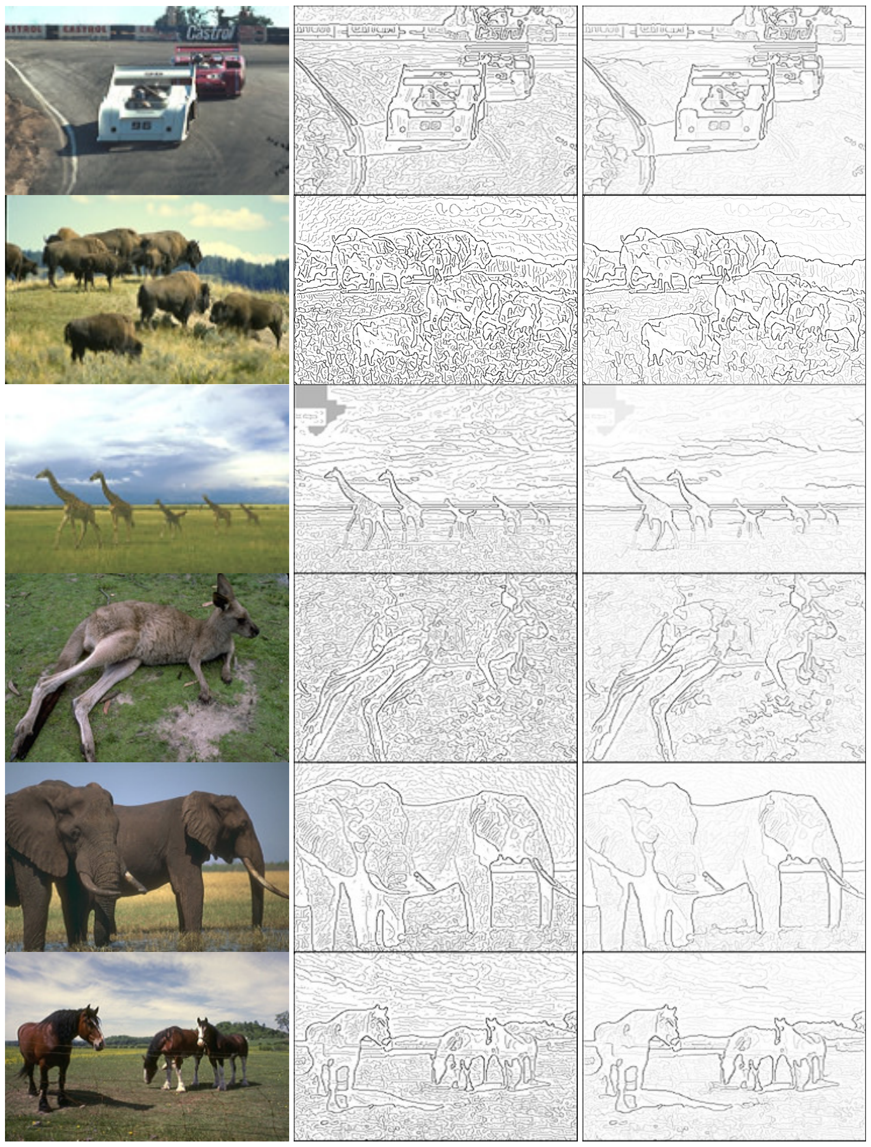
Quantitatively the improvement of boundary detection by torque is measured using 100 test images in the Berkeley dataset. While the Canny edge method scores 0.57, the torque-based strengthen edge method using the Canny edges increase the score to 0.59 in F-measure of the Berkeley benchmark.

Here, we demonstrate that state-of-art boundary detection algorithm, global Pb (gPb) [17], could be improved by using the torque operator. The approach is the similar to the one for improvement of visual attention. The gPb and the strengthened edges by torque (pbTorque) are blended by weighted sum, which is simply done with the following computation;

$$\text{gPbTorque} = (1-\alpha)\cdot \text{gPb} + \alpha \cdot \text{pbTorque}, \tag{3.26}$$

where α is a parameter for weight of blending. Both gPb and pbTorque represent boundary probability maps, where probability of boundary at each pixel are represented with value between 0 and 1. The weighted sum is computed for each pixel to generate a new boundary probability map gPbTorque. pbTorque is not independent from gPb because the edge map of gPb is used to compute pbTorque.

One approach for computing torque in this experiment was using gPb directly for torque computation, which means that torque value is weighted by the probability value in gPb. This approach was tested with the following values of



(a) (b) (c)

Figure 3.16: Examples of Strengthened Edges. (a) Test images. (b) Canny edges. (c) Strengthened edges. The Canny edges shown in (b) are used to compute strengthened edges in (c).

$\alpha \in \{0.05, 0.1, 0.2, 0.4\}$. The other approach was binalizing gPb before computing torque. In the following experiment, 0.1 is used to threshold the gPb, which represent boundary probability in the scale 0 to 1. For this approach, following values of $\alpha \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0\}$ were tested. Disk patches were used for this approach.

Randomly selected 24 images in Berkeley dataset were used for this experiment. The Table 3.4 shows the quality comparison in F-measure of the boundary detection in the order from the highest score. In the Algorithm column, App.1 and App.2 indicate either the first approach or second approach explained above, and the number after gPb+Torque shows the value of the parameter α. This table is visualized in the Fig. 3.17 as the changes over the weight parameter α so that we can see that boundary detection quality of gPb+Torque is better than gPb in the wide range of the parameter α between 0.1 and 0.6. The Fig. 3.18 shows the ROC curves for the best algorithm in the Table 3.4 and gPb. One plot for each ROC curve is marked with a bigger circle than other plots, at which the maximum F-measure scores shown in the Table 3.4 was obtained. The first approach using boundary probability value was expected to produce better result than the second approach because torque could take advantage of probability based on other visual cues. However, it is revealed that the second approach outperform the first one. It can be understood as follows: the binalization process prevent from transferring wrong assignment of boundary probability to torque computation. Therefore, the torque operator could spot points surrounded by structured edges even if small probabilities are assigned. In other words, due to the weighting based on the boundary probability, performance of torque operator is drawn by the quality of boundary probability map, and it was not possible to outperform the original boundary detection method by the first approach. On the other hand, the second approach could be understood that it made the torque operator most using cleaner binary edge maps. Regardless of probability values by gPb, the torque operator could generate additional mid-level cues to improve boundary detection performance.

Table 3.1: Boundary Detection Comparison with gPb. App.1 in Algorithm column represent the first approach where boundary probability values by gPb [17] are used to weight torque values. App.2 represent the second approach where the boundary probability map by gPb is binalized by threshold, and then torque is computed based on the binary edge maps.

Algorithm	F-measure
App.2 gPb+Torque 0.2	0.704
App.2 gPb+Torque 0.1	0.703
App.2 gPb+Torque 0.5	0.703
App.2 gPb+Torque 0.3	0.703
App.2 gPb+Torque 0.05	0.703
App.2 gPb+Torque 0.4	0.703
App.2 gPb+Torque 0.6	0.702
gPb	0.702
App.2 gPb+Torque 0.8	0.702
App.1 gPb+Torque 0.05	0.700
App.1 gPb+Torque 0.1	0.696
App.1 gPb+Torque 0.2	0.683
App.2 gPb+Torque 1.0	0.677
App.1 gPb+Torque 0.4	0.636

We found that quantitative measure of boundary detection performance sometimes depending on the type of database. In the previous experiment, the Berkeley image dataset [63] were used. The dataset mostly consists of nature images and not

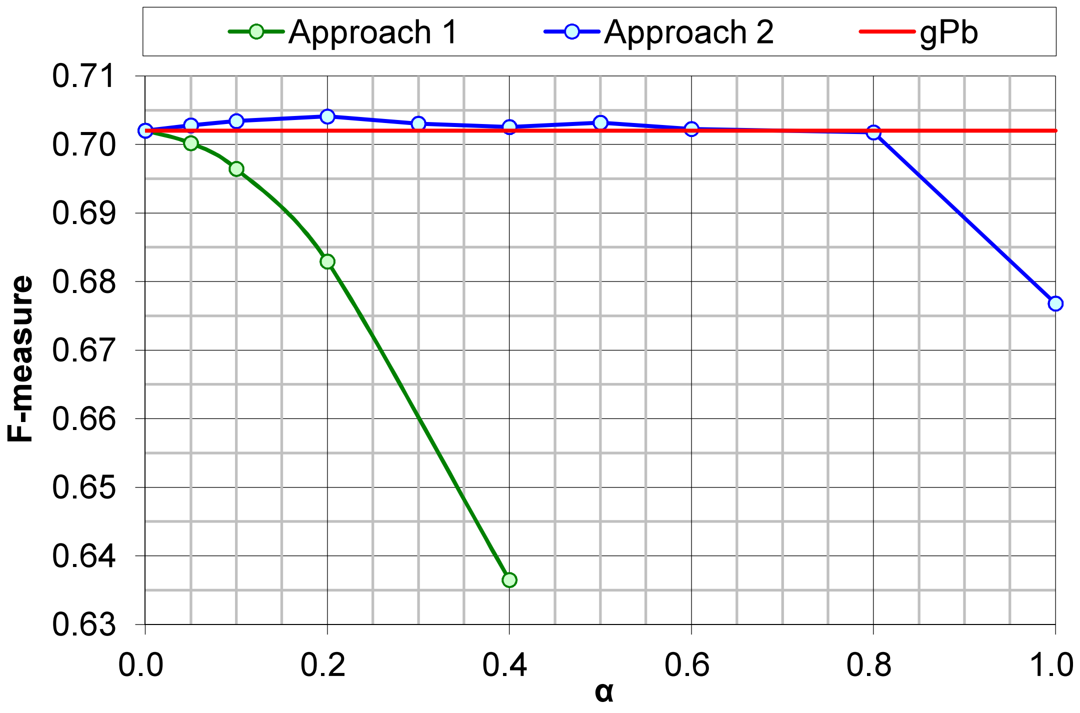


Figure 3.17: Boundary Detection Comparison over Parameter Changes. Approach 1 represent the first approach where boundary probability values by gPb [17] are used to weight torque values. Approach 2 represent the second approach where the boundary probability map by gPb is binalized by threshold, and then torque is computed based on the binary edge maps.

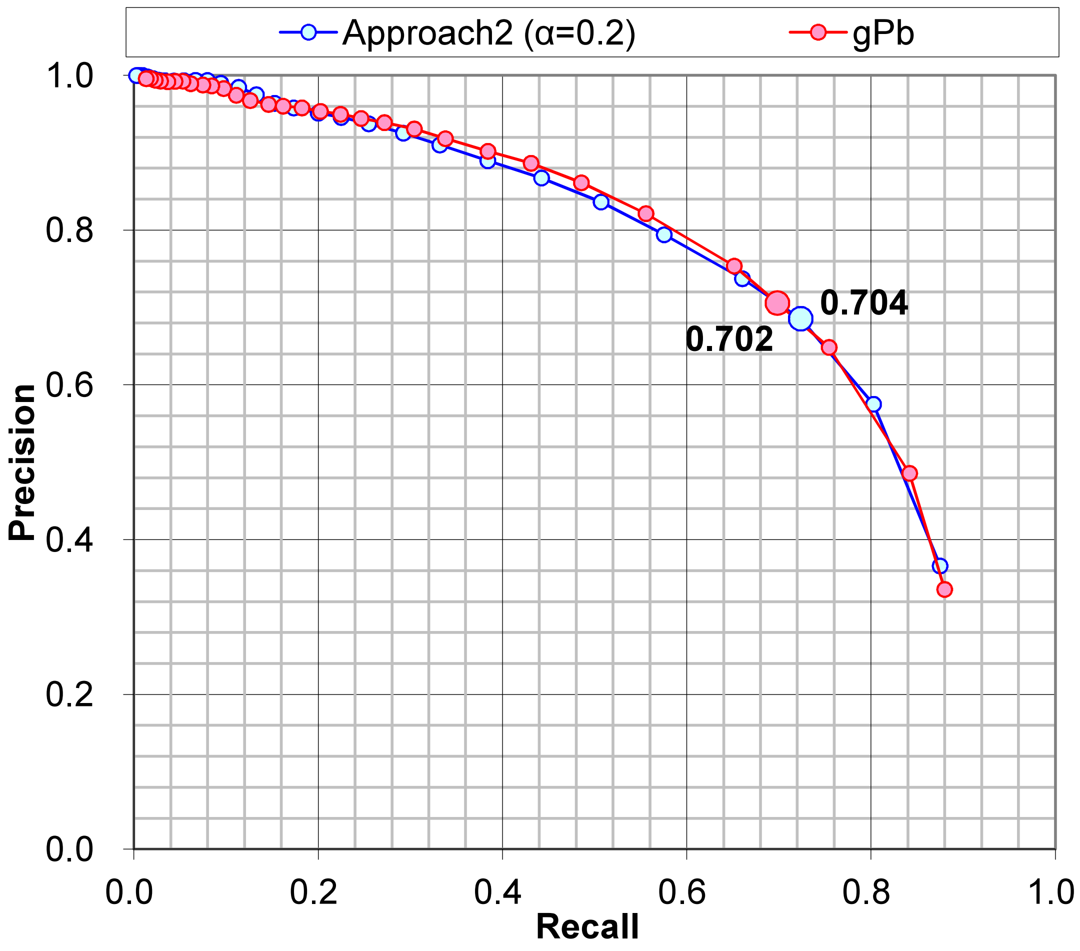


Figure 3.18: Boundary Detection Comparison with gPb. Precision-recall curve is shown. The boundary detection method by the approach 2 with the weight parameter 0.2 is compared with global Pb (gPb) method. The maximum F-measure is shown with a bigger marker of plot for each boundary detection method.

many object-oriented images. Therefore, the ground-truth boundaries varies over human subjects because boundaries in nature images often depends on the detail level of human subject's recognition of boundaries. We tried to check performance of boundary detection using torque operator with object-oriented dataset. Caltech dataset [66] was used in the next experiment. The improvement of boundary detection was 0.002 (0.2%) in F-measure in the previous experiment, but the improvement was 0.01-0.04 (1-4%) in the following experiment with the Caltech dataset.

In the following experiments in this section, we used 123 images in the 'car_side' category in the Caltech dataset. The category was chosen because of enough large number of images in dataset, relatively simple boundary, and reasonable size of object in images. Figure 3.19 shows samples of these test images and annotations.

For the edge strengthening, the idea is that edges which contribute to extrema more are strengthened more, but there is a question on how many extrema should be used. The strengthened edges are normalized between zero to one to represent probability map of boundaries. Therefore, intuitive expectation was that important edges are relatively less emphasized if too many edges are strengthened by too many extrema. It is expected that there is the reasonable number of extrema to be used for this process. In the following experiment, the relationship between number of extrema and performance of boundary detection is examined.

Boundary detection by edge strengthening are performed with different numbers of extrema used in the edge strengthen process. In this experiment, Canny edge [67], Berkeley edge (pb) [63] and global Pb (gPb) [17] were used as base edge maps. The strengthened edges were computed for each of these base edge maps, and final boundary probability maps were computed by weighted sum between the base edge map and the strengthened edge map. The weight were varied in the experiment. Figure 3.20, 3.21, and 3.22 show examples of boundary detections for Canny edges, pb edges, and gPb edges respectively. The boundary probability maps in each row are generated using the weight value shown in the first column. For these figures, 5000 extrema are used, but in the experiment the number of extrema used for the edge strengthening process were varied in 10, 50, 100, 500, 1000 and 5000. Figure 3.23 shows the plots of maximum F-measure values over number of extrema and weight value for the weighted sum. The precision-recall curves for each base edge map, each number of extrema, and each weight parameter value is shown in Fig. 3.24 and Fig. 3.25.

The maximum F-measure values over number of extrema and weight parameter value are summarized in Table 3.4. In consequence, performance was not improved without blending with the base edge map, which can be seen in the corresponding plots of weight 1.00. Using more than 100 extrema, performance was improved with the weight 0.25 or 0.50 than the base edge maps. The performance tends to be higher according to number of extrema with the weight 0.25 or 0.50. Using 5000 extrema with weight 0.5 gave the best performance for all Canny, pb and gPb edge maps.

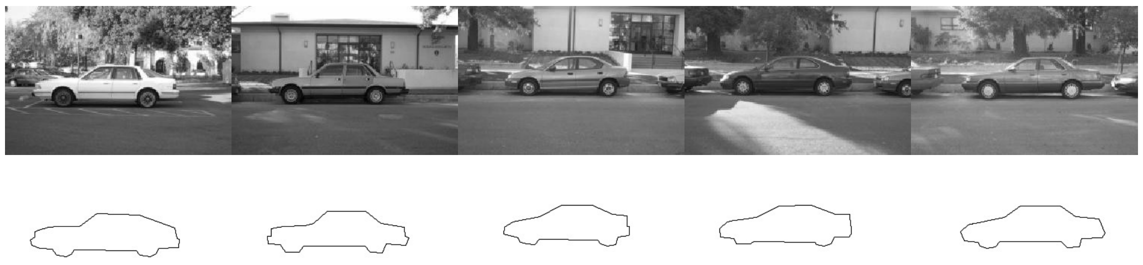


Figure 3.19: Example of Test Images and Annotations: Five example images and corresponding annotations in 'car_side' category in Caltech dataset [66] are shown.

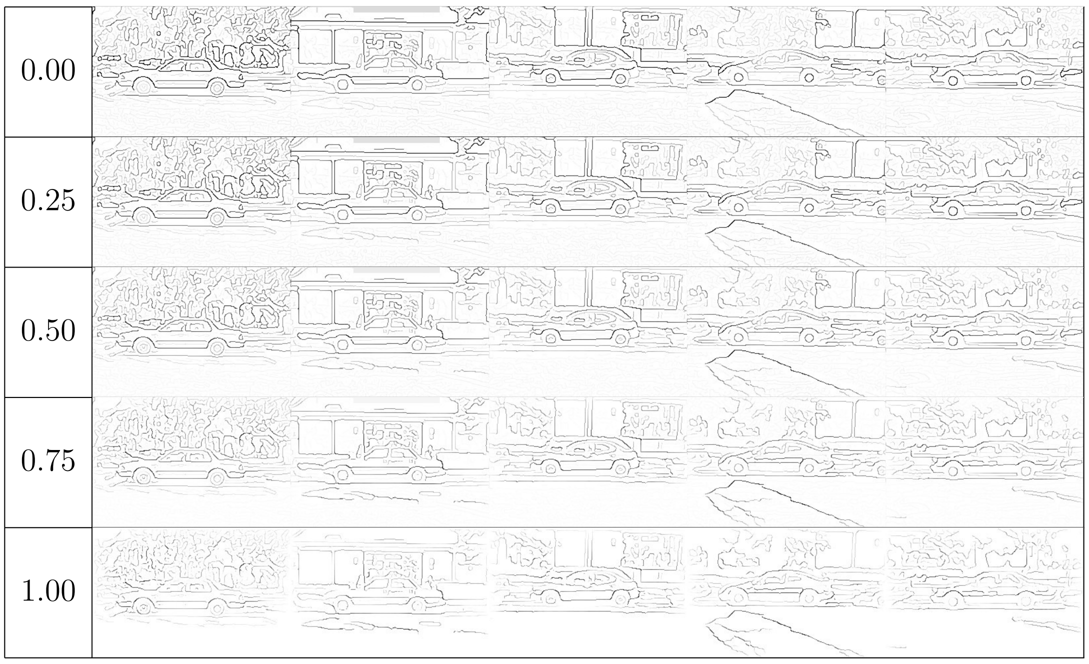


Figure 3.20: Example of Boundary Detection using Canny Edges

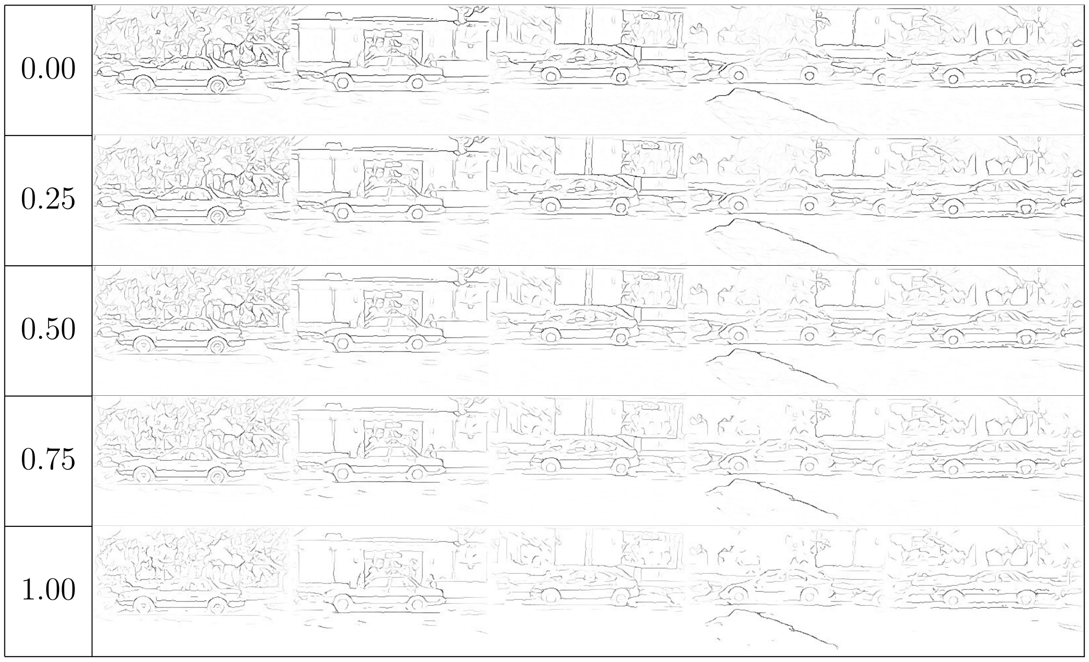


Figure 3.21: Example of Boundary Detection using pb Edges

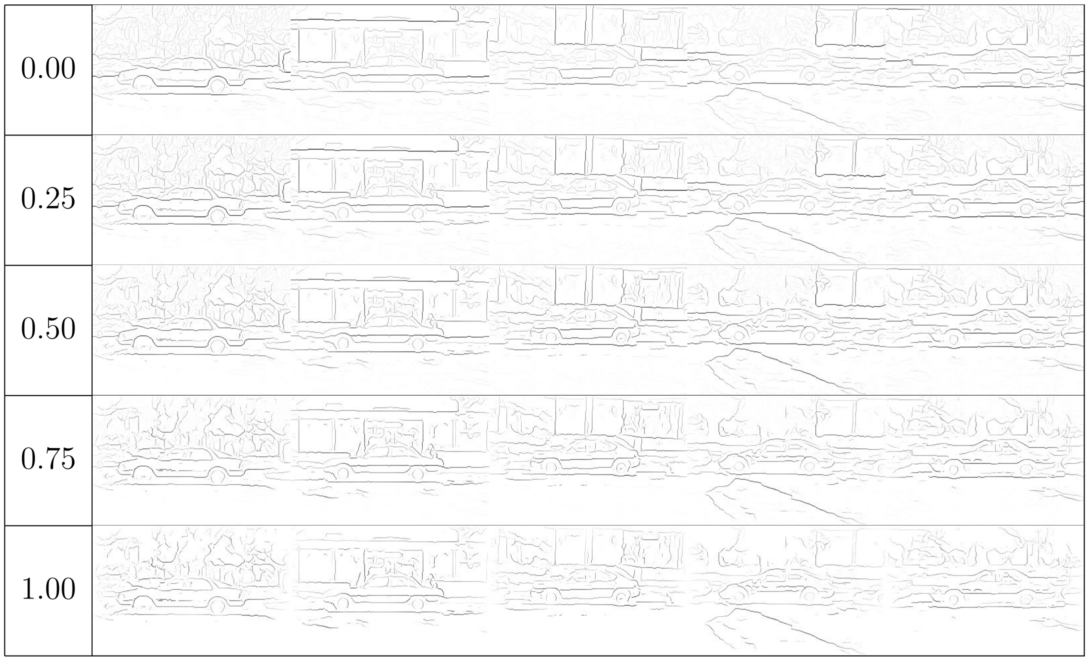
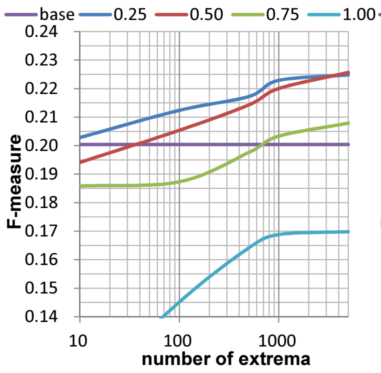
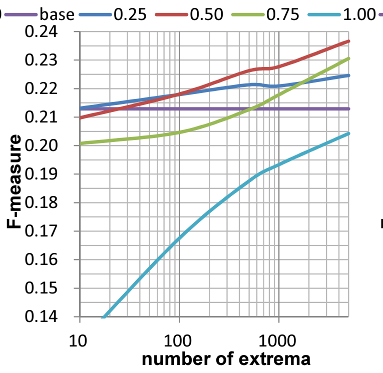


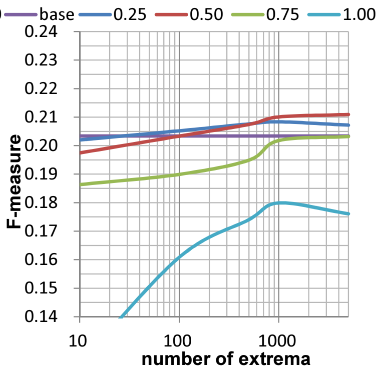
Figure 3.22: Example of Boundary Detection using gPb



(a) Canny

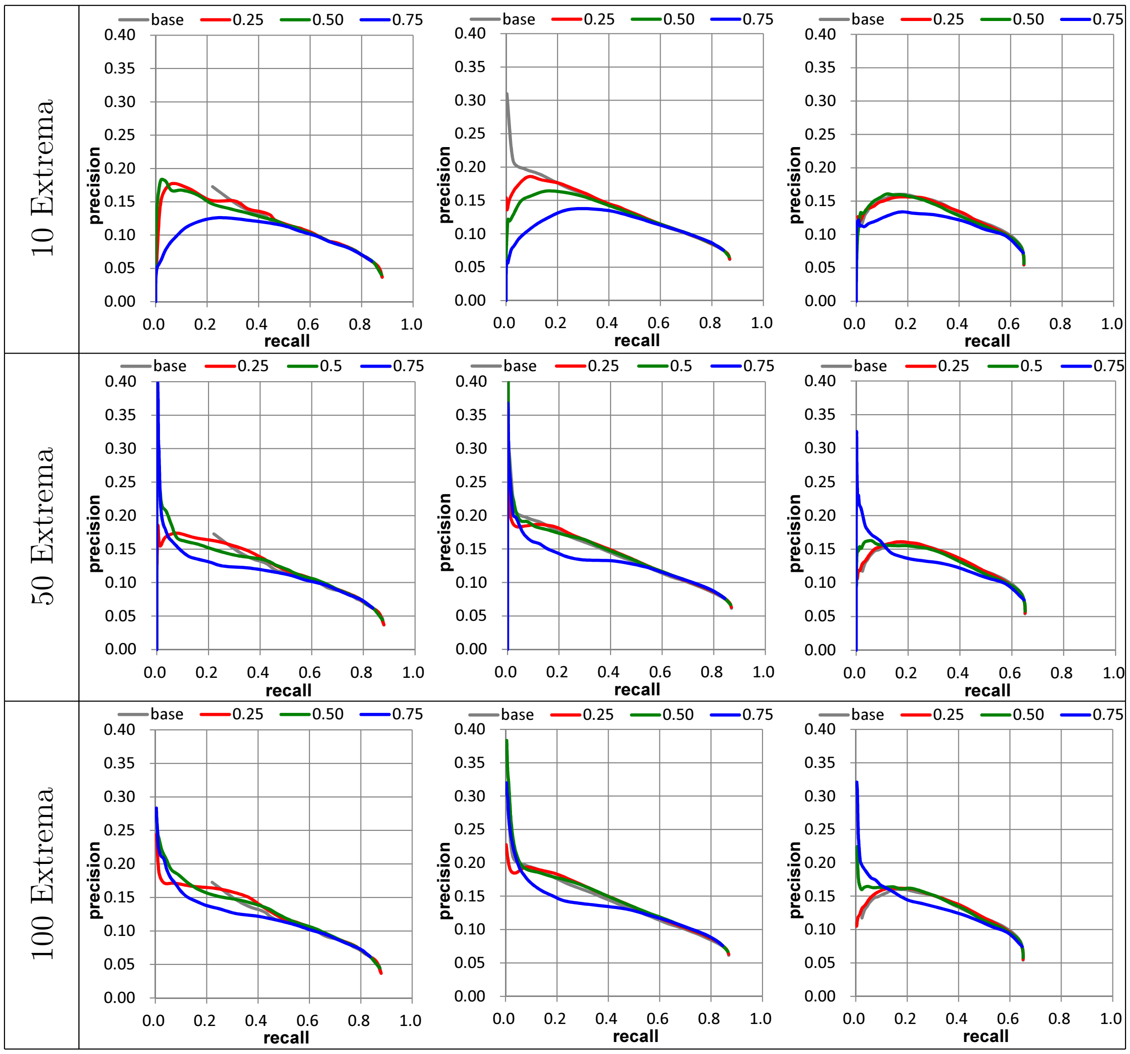


(b) pb



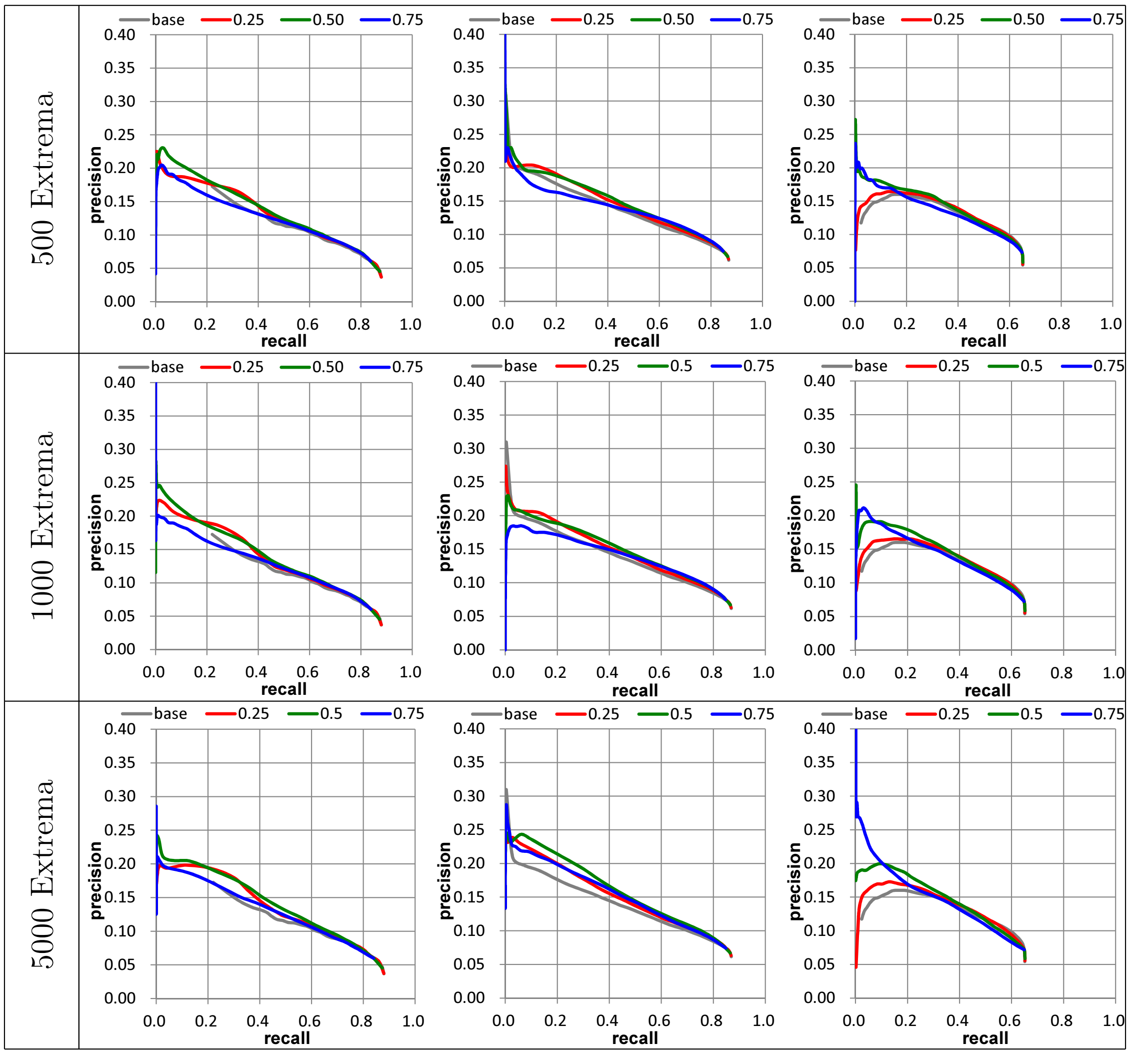
(c) gPb

Figure 3.23: Performance of Boundary Detection and Number of Extrema. Following edge detection method were used as base edges to compute torque and blended with the computed torque-based edge map: (a) Canny edges, (b) pb edges, and (c) gPb edges. The 'base' in legend means the base edge maps without edge strengthening. The numbers in legend represent weights for weighted sum of the base edge and strengthened edges by torque operator for generating boundary probability maps.



(a) Canny (b) pb (c) gPb

Figure 3.24: Precision and Recall Curve for Boundary Detection and Weight Parameters 1/2. Following edge detection method were used as base edges to compute torque and blended with the computed torque-based edge map: (a) Canny edges, (b) pb edges, and (c) gPb edges. The 'base' in legend means the base edge maps without edge strengthening. The numbers in legend represent weights for weighted sum of the base edge and strengthened edges by torque operator for generating boundary probability maps.



(a) Canny (b) pb (c) gPb

Figure 3.25: Precision and Recall Curve for Boundary Detection and Weight Parameters 2/2. Following edge detection method were used as base edges to compute torque and blended with the computed torque-based edge map: (a) Canny edges, (b) pb edges, and (c) gPb edges. The 'base' in legend means the base edge maps without edge strengthening. The numbers in legend represent weights for weighted sum of the base edge and strengthened edges by torque operator for generating boundary probability maps.

Table 3.2: Number of Extrema and Performance of Boundary Detection: F-measure for each type of base edges, weight parameter value, and number of extrema used for edge strengthening is shown.

		weight				
base edges	extrema	0.00	0.25	0.50	0.75	1.00
Canny	10	0.20	0.20	0.19	0.19	0.11
	50	0.20	0.21	0.20	0.18	0.14
	100	0.20	0.21	0.21	0.19	0.15
	500	0.20	0.22	0.21	0.20	0.16
	1000	0.20	0.22	0.22	0.20	0.17
	5000	0.20	0.22	**0.23**	0.21	0.17
pb	10	0.21	0.21	0.21	0.20	0.13
	50	0.21	0.22	0.22	0.20	0.16
	100	0.21	0.22	0.22	0.20	0.17
	500	0.21	0.22	0.23	0.21	0.19
	1000	0.21	0.22	0.23	0.22	0.19
	5000	0.21	0.22	**0.24**	0.23	0.20
gPb	10	0.20	0.20	0.20	0.19	0.12
	50	0.20	0.20	0.20	0.19	0.15
	100	0.20	**0.21**	0.20	0.19	0.16
	500	0.20	**0.21**	**0.21**	0.19	0.17
	1000	0.20	**0.21**	**0.21**	0.20	0.18
	5000	0.20	**0.21**	**0.21**	0.20	0.18

3.5 Segmentation

There are many approaches to image segmentation, but we discuss here on object segmentation in images. This problem is also know as foreground-background segmentation or figure-ground segmentation. As we discussed in the visual attention section (Sec. 3.3), points to draw attention could be specified using torque operator and other visual cues. Therefore, after we check possibility of segmentation without such an attention point, we take an approach of fixation based segmentation method [37] by assuming attention points or fixation points are given. Optimization approach gains popularity in image segmentation because of good performance. The fixation based segmentation approach proposed by Mishra *et al.* is also categorized into optimization approach, but in contrast to other methods in this category, image space is converted to polar coordinate from Cartesian coordinate where the fixation point is used as the origin of the polar coordinate system. The segmentation is obtained by optimization so that an image is segmented into an object region containing the fixation point and rest of image region as background. The segmented object boundary is optimized to match mostly to detected edges. Because of conversion to polar coordinate, the object segmentation is basically obtained by splitting an image into half by a line parallel to the axis of angle. We take this concept for utilizing torque operator in segmentation.

3.5.1 Segmentation using Torque Value Map

We tried to perform segmentation of objects in images based on the torque value map using graph-cut algorithm. The torque value map is utilized in the unary term for the graph-cut by assuming that the higher (lower) torque is the higher (lower) the chance to be a foreground object because we have observed that torque values inside an object region are mostly in the same sign.

First, we used only unary term (data cost) E_d of cost function in graph-cut. The data cost is computed as follows:

$$E_d\left(L\right) = \sum_{x,y} E\left(x,y,l\right), \tag{3.27}$$

$$E_d\left(x,y,l\right) = \begin{cases} \tilde{\tau}\left(x,y\right) & l=0, \\ \\ 1-\tau\left(x,y\right) & l=1, \end{cases} \tag{3.28}$$

where $\tilde{\tau}\left(x,y\right) \in [0,1]$ is linearly normalized torque value at (x,y). $l \in \{0,1\}$ is a label. L is set of labels representing segmentation. Fig. 3.26 shows original image in the left column, segmentation result in the middle, and data cost for graph-cut.

Second, we added binary term (smoothness cost) E_s based on color edges,

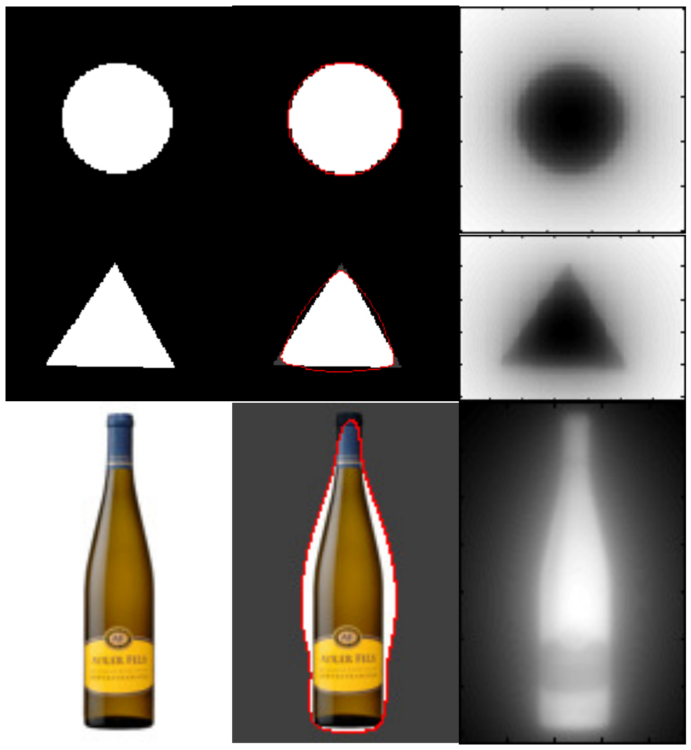

(a) (b) (c)

Figure 3.26: Segmentation using only Torque. (a) Test image. (b) Segmentation. (c) Data term cost in cost function.

which is defined as follows:

$$E_s(L) = \sum_{(p,q)\in\mathcal{N}} E_s(l_p, l_q), \tag{3.29}$$

$$E_s(l_p, l_q) = \begin{cases} 0 & l_p = l_q, \\ 1 - \Delta c_h(p) & l_p \neq l_q, (p,q) \in \mathcal{N}_h, \\ 1 - \Delta c_v(p) & l_p \neq l_q, (p,q) \in \mathcal{N}_v, \end{cases} \tag{3.30}$$

$$\Delta c_h(x, y) = \frac{1}{3} \sum_{c=\{r,g,b\}} |I(x+1, y, c) - I(x, y, c)|, \tag{3.31}$$

$$\Delta c_v(x, y) = \frac{1}{3} \sum_{c=\{r,g,b\}} |I(x, y+1, c) - I(x, y, c)|, \tag{3.32}$$

where l_p is a label assigned at point p. $I(x, y, c)$ is intensity at point (x, y) in color channel c. Then, the cost function is defined as follows:

$$E(L) = E_d(L) + \alpha \cdot E_s(L). \tag{3.33}$$

Fig. 3.27 shows some results of the graph-cut using both data cost and smoothness cost. The first column shows images, the second column shows segmentations, and the third and fourth column show data cost and smoothness cost respectively. In this experiment, $\alpha = 1.5$ was used.

Obviously, by adding smoothness term based on color similarity, better segmentations than using only data cost in the sense of fitting to its boundary are obtained. From these experiments, we confirmed that the torque value map is useful to obtain the rough information on object location and size, but edges based on either color or intensity is required for the accurate boundary location.

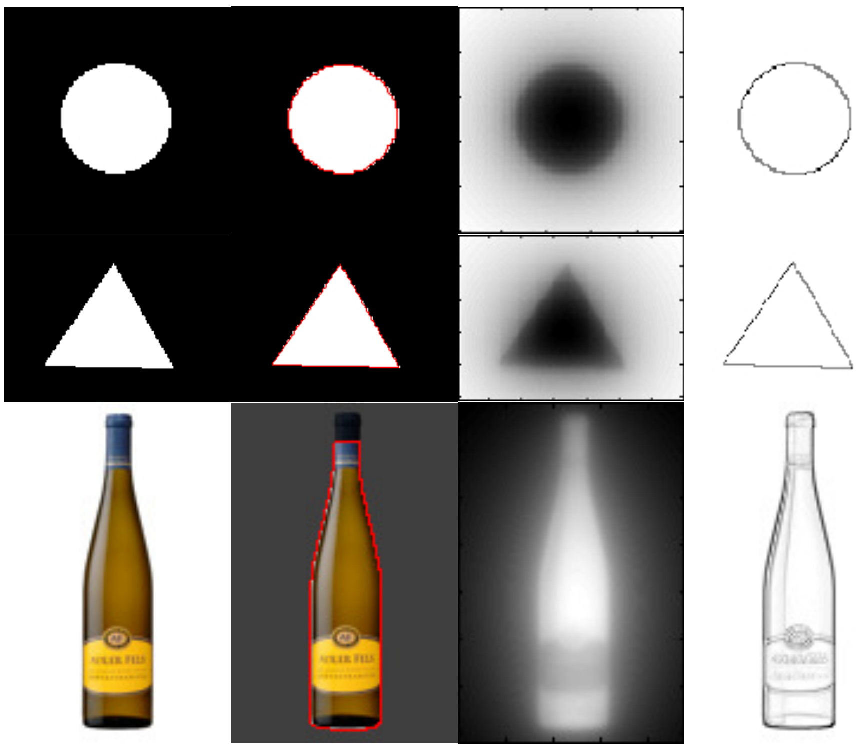

(a) (b) (c) (d)

Figure 3.27: Segmentation using Torque Map and Color. (a) Test image. (b) Segmentation. (c) Data term cost in cost function. (d) Smoothness term cost in cost function.

3.5.2 Segmentation using Torque Value on Edges

A intuitive method for utilizing torque concept in fixation based segmentation is using torque value for each edge point directly in the optimization process of segmentation. In this method, edges used for torque computation are converted into polar coordinate system for segmentation. Since a fixation point is given, torque is computed regarding to the given fixation point as the center of rotation in the following experiment. Let us denote an arm vector from the fixation point to an edge pixel by $\vec{r}$, and an edge vector at the end point of $\vec{r}$ by $\vec{v}$. The edge vector, $\vec{v}$, can be decomposed into the component along the arm vector $\vec{r}$, and the other component perpendicular to $\vec{r}$ as depicted in Fig. 3.28. Then a magnitude of torque associating with the edge vector can be simply written as follows:

$$\tau = \|\vec{r} \times \vec{v}\| = r \cdot v_\theta, \tag{3.34}$$

where $r = \|\vec{r}\|$, and θ is the angle between $\vec{r}$ and $\vec{v}$. Therefore, once an image is transformed into polar coordinate, the map of edge vectors can be represented by v_r, components parallel to the axis of distance r, and v_θ, components parallel to the axis of angle θ in the polar coordinate system as in Fig. 3.29.

In this experiment, we used edges defined based on gradient as $v = (-I_y, I_x)$, then v_θ is written as follows:

$$v_\theta = (I_x \sin\theta - I_y \cos\theta), \tag{3.35}$$

where I_x, I_y are x and y component of image gradient respectively. We normalized $\vec{r}$ to be the unit vector in this experiment so that distance from the fixation point to

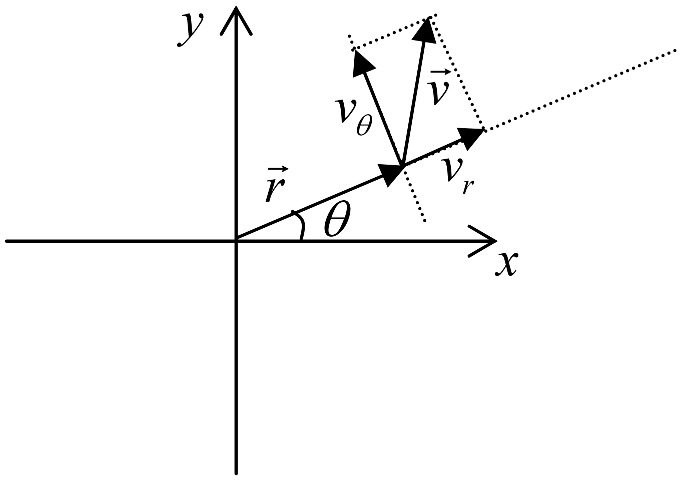


Figure 3.28: Decomposition of Edge Vector. An edge vector $\vec{v}$ at the end point of an arm vector $\vec{r}$ can be decomposed into the component along the arm vector $\vec{v_r}$ and the component perpendicular to the arm vector $\vec{r_\theta}$.

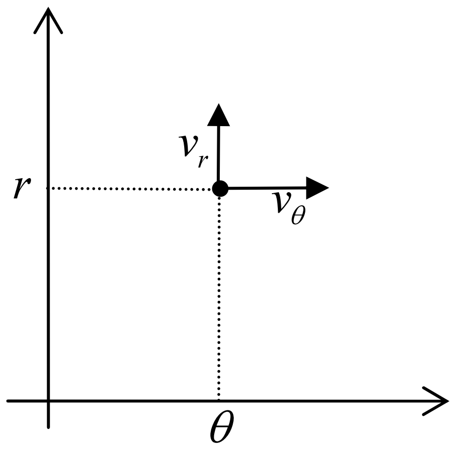


Figure 3.29: Representation of Edge Vector in Polar Coordinate System. Edge vectors are represented by v_r, the component parallel to the axis of distance r, and v_θ, components parallel to the axis of angle θ.

the edge does not bias the segmentation. Furthermore, we only use the magnitude of torque because both bright-to-dark edge and dart-to-bright edge could be boundary edges. Then, the torque value can be simply computed as follows:

$$\tau = v_\theta = \left| I_x \sin\theta - I_y \cos\theta \right|. \tag{3.36}$$

The sum of v_θ in a rectangular region $r < r_0$ in the polar coordinate corresponds to a torque over a circular patch of radius r_0.

Based on the above formulas, torque value for each edge pixel regarding to the origin of coordinate system is computed. The origin of coordinate system is chosen to be the given fixation point in this experiment. The fixation point may be one of local extrema in torque map. For example, polar coordinate image is shown in Fig. 3.30, and edge vector components are shown in Fig. 3.31 in brightness coding. Please note that angular component of an edge is equivalent to torque by definition in this experiment.

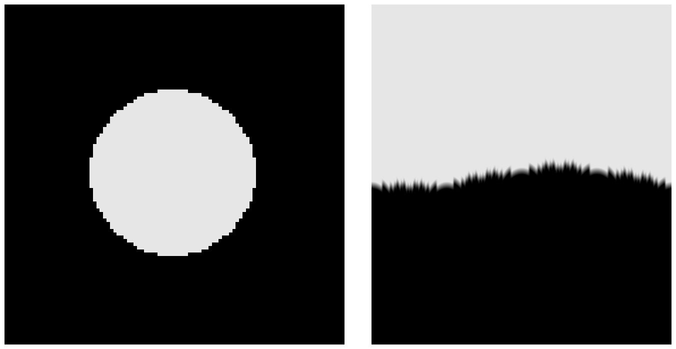


Figure 3.30: Images in Cartesian and Polar Coordinate System: Both left and right images shows the same shape in different representation by using different coordinate system. Left: Cartesian coordinate system. Right: Polar coordinate system.

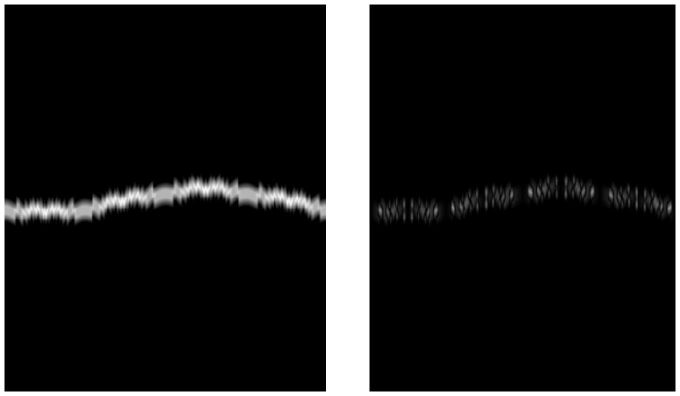


Figure 3.31: Edge Vector Components: Maps for edge components are shown for v_θ on the left and v_r on the right.

In this example, torque, which is equivalent to angular component of edge, is expected to be significantly large at the boundary of object, because it has most likely strong gradient and surrounding structure around the fixation point. Therefore, we used the torque values in the smoothness term of cost function, and then segmentation which minimize the cost function was obtained by the graph-cut method. A graph depicted in Fig. 3.32 was constructed for this segmentation. The nodes of s and d in the figure are source and drain nodes respectively, and other nodes correspond to pixels in polar coordinate. The minimization of the cost function is equivalent to dividing the graph into two subgraphs, one containing the source node and the other containing the drain node, where the sum of weights associating with the edge of the cut is the minimum. The pair wise smoothness term represents the weight on the edge between connected two nodes, and it is defined as follows:

$$w_{pq} = \exp\left(-\alpha \sum_{p' \in \mathcal{N}_p} \tau_{p'}\right), \tag{3.37}$$

where α is a positive constant parameter. $\mathcal{N}_p$ is a set of neighboring pixels of p. τ_p

is torque value at p. Since the weight is smaller when the torque value is larger by the definition, it is expected that the graph is cut at edges of high torque. Since the circular connection for axis of angle is structured in the graph, one segment containing the fixation point with closed boundary in Cartesian coordinate system is guaranteed after the cut.

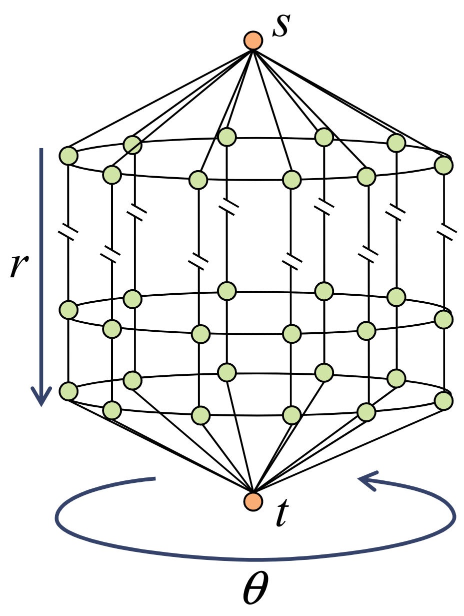


Figure 3.32: Structure of Graph for Segmentation.

Some examples of resulting segmentation are shown in Fig. 3.33 and Fig. 3.34. The first column shows original images with a given fixation points. The second column shows polar coordinate images regarding to the fixation points. Smoothness cost based on torque value in polar coordinate system is shown by brightness coding in the third column. The fourth column shows segmentation by graph-cut. The red line represents a cut on a graph. The fifth column shows segmentation results, where the resulting segmentation is delineated by close boundary in red. The boundaries correspond to the graph-cut in the fourth column. The circular regions in fifth column are maximum circles inside images centered at the given fixation points. The polar coordinate images are generated only for the circular regions.

Reasonable results are obtained for the first four example test images, but segmentations for rest of images does not look visually good. It can be seen often that background edges disturb the segmentation process. In this experiment, we used torque value at each edge point, but did not take the advantage of torque operator to represent existences of structured surrounding edges. In the next approach, we improve segmentation using the strengthened edges by the torque operator, with which we will take structured edges into account through the torque operator.

3.5.3 Strengthened Edges

The strengthened edges are expected to be useful for foreground segmentation because object boundaries are emphasized. There are many possibilities to utilize the strengthened edges, but we show here figure-ground segmentation using the strengthened edges.

For a quantitative evaluation of figure-ground segmentation, the dataset by Stein *et al*. [68] was used in this experiment. We selected single foreground object for each reference image from the dataset, and the centroid of the object in the image is used as a fixation point given for segmentation. We applied graph-cut segmentation in polar coordinate to the test images using the fixation points [37]. Different visual cues are used for the graph-cut segmentation for comparison; Canny

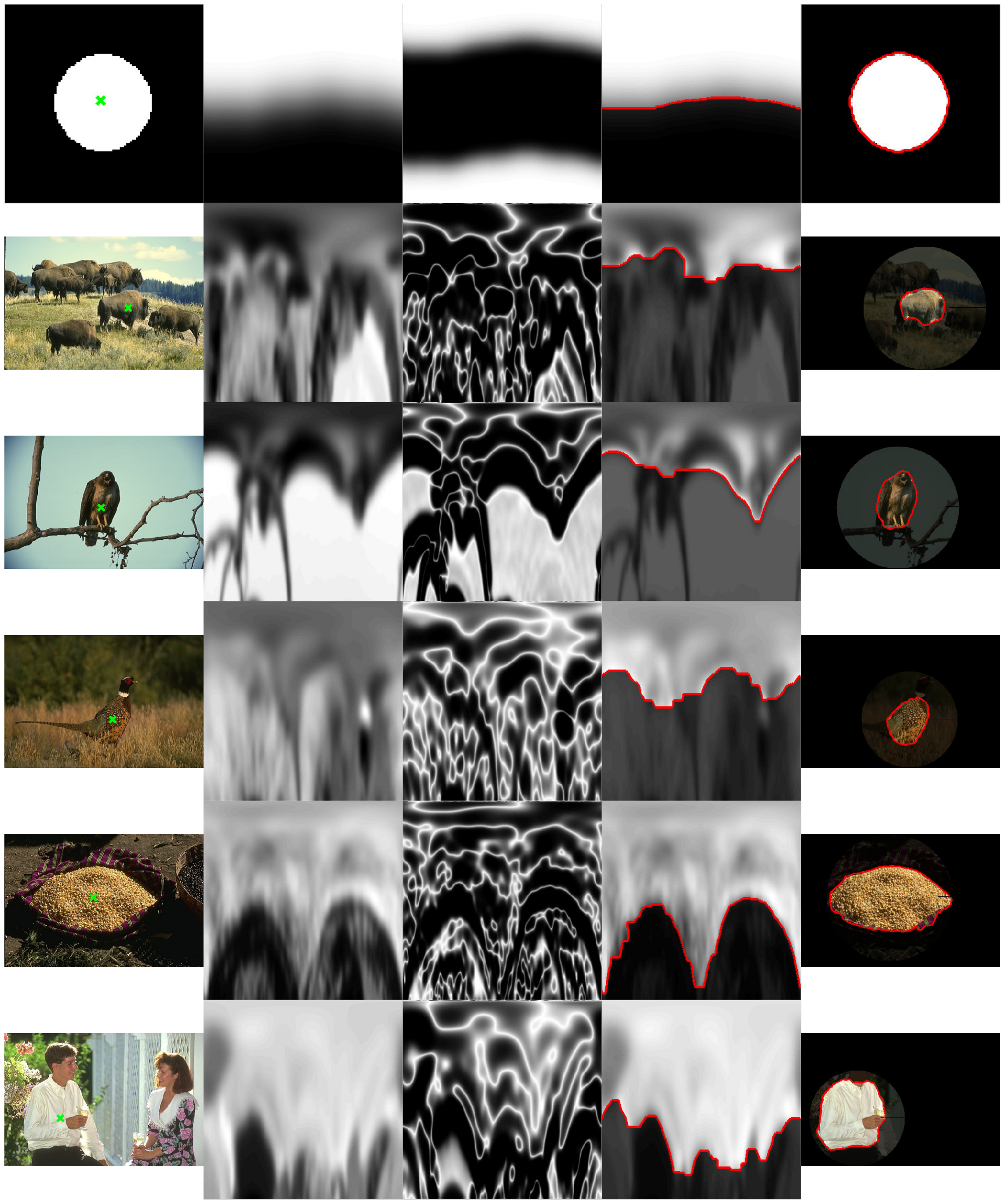


Figure 3.33: Fixation-based Segmentation using Torque Value on Edges 1/2: (a) Test image with fixation point marked by green color x. (b) Test image converted to polar coordinate system regarding to the fixation point. (c) Smoothness cost for optimization. The brighter means the higher cost. (d) Segmentation in polar coordinate system. (e) Segmentation in Cartesian coordinate system.

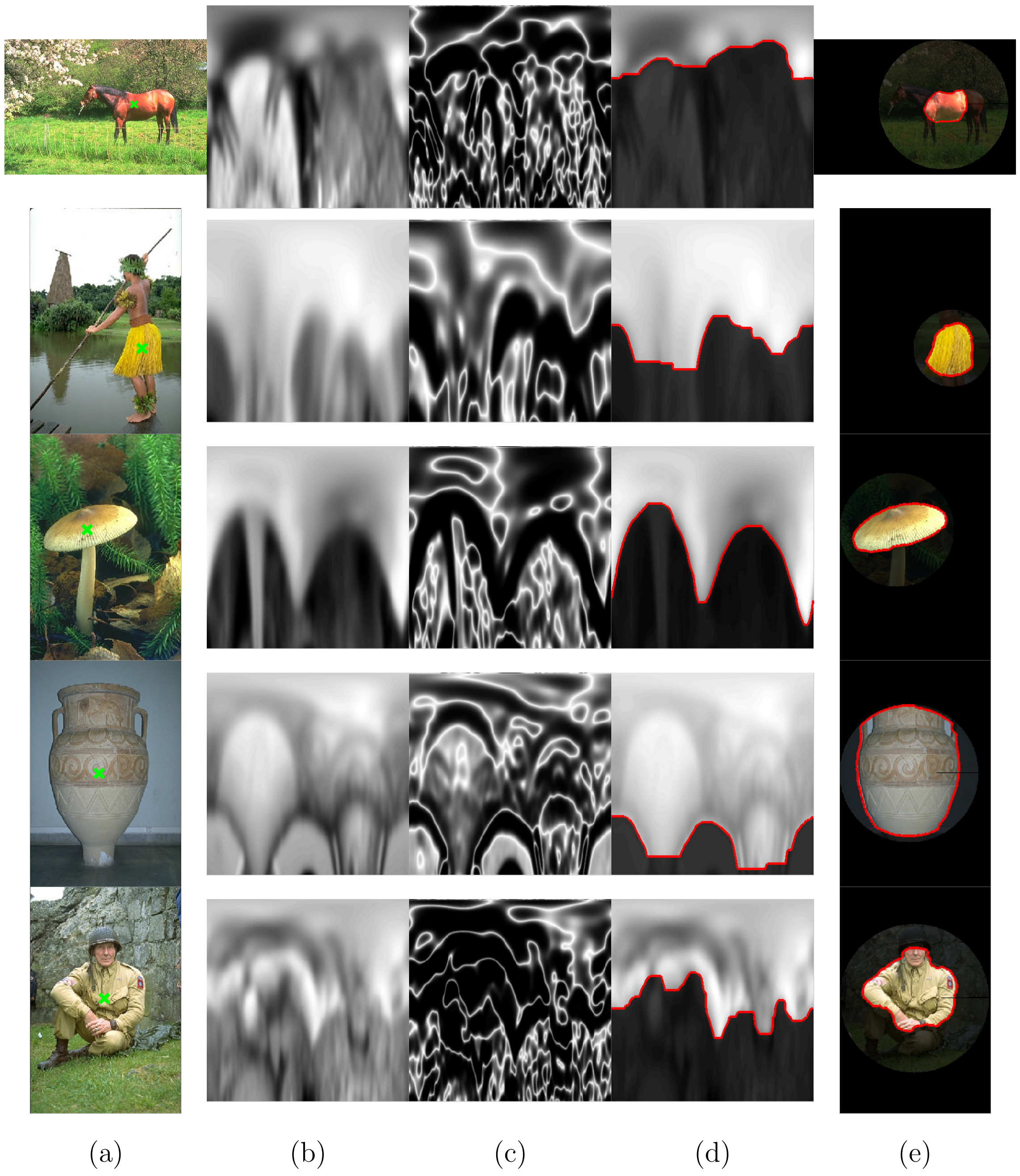


Figure 3.34: Fixation-based Segmentation using Torque Value on Edges 2/2: (a) Test image with fixation point marked by green color x. (b) Test image converted to polar coordinate system regarding to the fixation point. (c) Smoothness cost for optimization. The brighter means the higher cost. (d) Segmentation in polar coordinate system. (e) Segmentation in Cartesian coordinate system.

edge map, boundary probability map (pb) by Martin *et al.* [15], strengthened edge map using Canny edges, and strengthened edge map using Pb edges. In addition, we also compared with the non-edge based level-set segmentation by Chan and Vese [64]. A small rectangular region centered at the given fixation point was used as seed region in this method. In order to see the effect of torque measure directly, the normalized torque contribution d_t in eq. (3.25) is used as strengthened edge in this experiment. The quality of segmentation is evaluated by segmentation covering defined as follows [69];

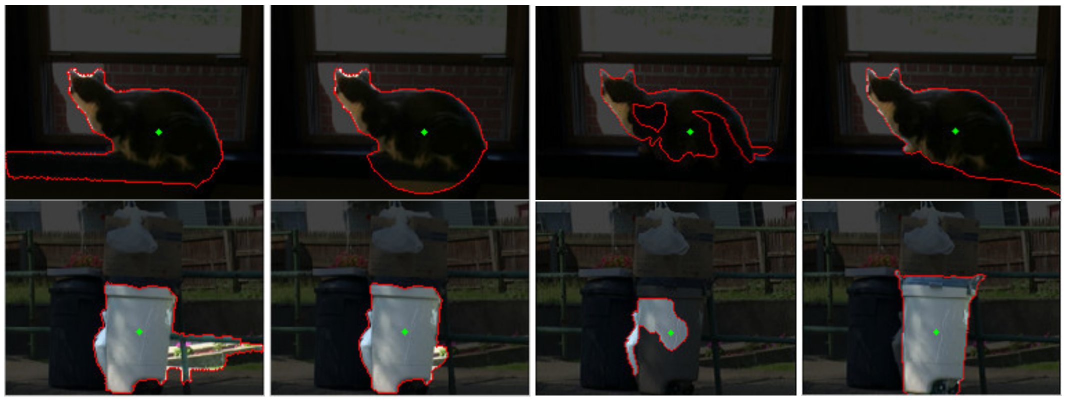
$$C = \frac{|S \cap \mathcal{G}|}{|S \cup \mathcal{G}|}, \tag{3.38}$$

where S and $\mathcal{G}$ are the computed segmentation and ground truth segmentation respectively. Both S and $\mathcal{G}$ are represented by binary labeling.

Table 3.5.3 shows quantitative comparison on the dataset. Average covering over valid 28 test images for each visual cue is shown in the table. The 'Torque' in the table indicates segmentation using an edge map strengthened by the torque measure, using Canny in the left table and Pb in the right table. As seen in the table, segmentation quality is improved by introducing the torque measure as a mid-level visual cue comparing with the segmentation using the original edge maps by local cues. We can also see from the performance of the Chan-Vese method, that the segmentation of objects for this data set, given only the fixation point, is a challenging task. Examples of segmentation are shown in Fig. 3.35.

Table 3.3: Covering of Foreground Segmentation using Different Visual Cues: Torque (Canny) and Torque (pb) mean that strengthened edges by torque operator using Canny and pb edges as the base edge map is used as visual cue respectively. Comparing with the base edge map, Torque improves segmentation performance in covering both for Canny and pb edges. The low performance by Chan-Vese method [64] indicates that the segmentation for this dataset was challenging.

Visual Cue	Covering
Canny	0.32
Torque (Canny)	**0.47**
pb	0.40
Torque (pb)	**0.48**
Chan-Vese	0.21



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

Figure 3.35: Examples of Segmentations: (a) Segmentation using Canny edges. (b) Segmentation using strengthened edges by torque. (c) Segmentation by Chan-Vese method [64]. (d) Ground truth. The green dots are the fixation points.

3.5.4 Top-down Knowledge of Shape

We demonstrate here how segmentation can be adapted to top-down knowledge of shape. We assume that an elliptic region is given as a top-down knowledge of the object shape, and the segmentation by graph-cut in polar coordinate system is adapted to ellipse. It is worthwhile to note that the rough estimate of object shape as the top-down knowledge of shape might be obtained using torque as we see in connected components in Sec. 3.2.3. The graph-cut in polar coordinate system can be adapted to ellipse by modifying the mapping. The map between Cartesian coordinate (x, y) and polar coordinate (θ, r) is modified so that points on the ellipse in Cartesian coordinate are mapped onto a line of constant distance r in the polar coordinate as follows:

$$\begin{pmatrix} x \\ y \end{pmatrix} = R \cdot S \cdot r \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \tag{3.39}$$

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}, \tag{3.40}$$

$$S = \frac{1}{ab} \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}, \tag{3.41}$$

where a and b are the ratio of semi-major and semi-minor axis respectively. α is the angle of major axis to the x-axis in Cartesian coordinate.

We demonstrate this method for an image form the Berkeley image dataset. An ellipse close to an object shape was generated by manually adjusting parameters of ellipse. Segmentations in the polar coordinate using the parameters of ellipse

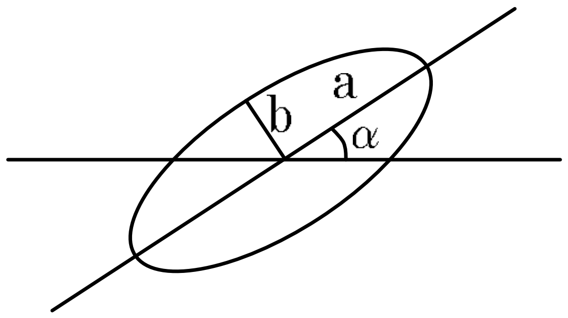


Figure 3.36: Parameters of Ellipse: a and b is major and minor radius respectively. α is the angle of major axis to the x-axis in Cartesian coordinate.

in eq. (3.41) was performed. By the transformation of eq. (3.39), elliptic region is transformed to a rectangular region in polar coordinate as shown in Fig. 3.37. For comparison, segmentation in the standard polar coordinate was also performed. Resulting segmentations are shown in Fig. 3.38. The top raw in the figure shows edges used for segmentation. The left most column shows ellipse used for segmentation. Fig. 3.38 (a) and (b) are Canny edge map and strengthened Canny edge map respectively. Fig. 3.38 (c) and (d) are Berkeley edge map and strengthened Berkeley edge map respectively. For Fig. 3.38 (i), ellipse adjusted for an object in the image is used. For comparison, standard polar coordinate is used for Fig. 3.38 (ii). It can be seen that segmentation can be adapted to the top-down knowledge of shape, and the segmentation quality is improved by using elliptic coordinate for strengthened Canny edge. It shows that better segmentation can be obtained by strengthened edges in comparison with the base edges, and by the top-down knowledge of the object shape in comparison with the segmentation without the top-down knowledge.

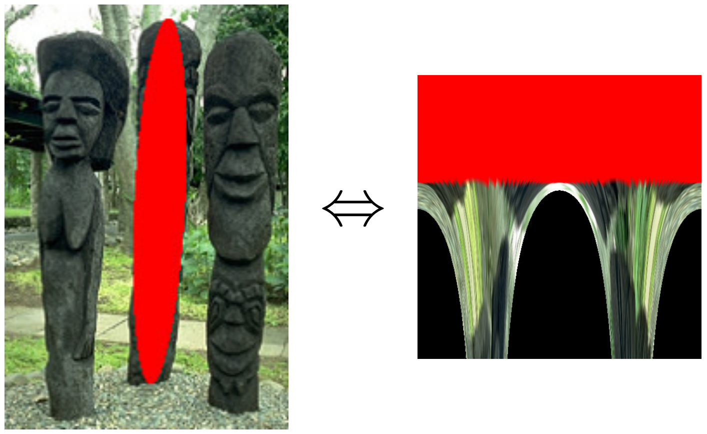


Figure 3.37: Mapping between Cartesian to Elliptic Polar Coordinate.

In the previous experiment, the image and edges are actually warped to elliptic polar coordinate system. However, this adaptation can be done by weighting smoothness term of cost function depending on the distance to neighboring pixels in the transfered coordinate system.

$$E\left(L\right)=\sum_{p}E_{d}\left(l_{p}\right)+\sum_{(p,q)\in\mathcal{N}}w\left(p,q\right)E_{s}\left(l_{p},l_{q}\right). \tag{3.42}$$

The weight w is in general adapted to arbitrary coordinate system defined by the top-down shape. For ellipses, the distance between two points in elliptic polar coordinate system can be used for the weight. The weight can be given for the elliptic polar coordinate system as follows:

$$w\left(p,q\right)=\sqrt{dr^{2}+d\theta^{2}}, \tag{3.43}$$

$$dr=\frac{x'dx'+y'dy'}{r}, \tag{3.44}$$

$$d\theta=\frac{1}{ab}\cdot\frac{-y'dx'+x'dy'}{r^{2}}, \tag{3.45}$$

where $x'=r\cos\theta$ and $y'=r\sin\theta$. a and b are the ratio of semi-major and semi-

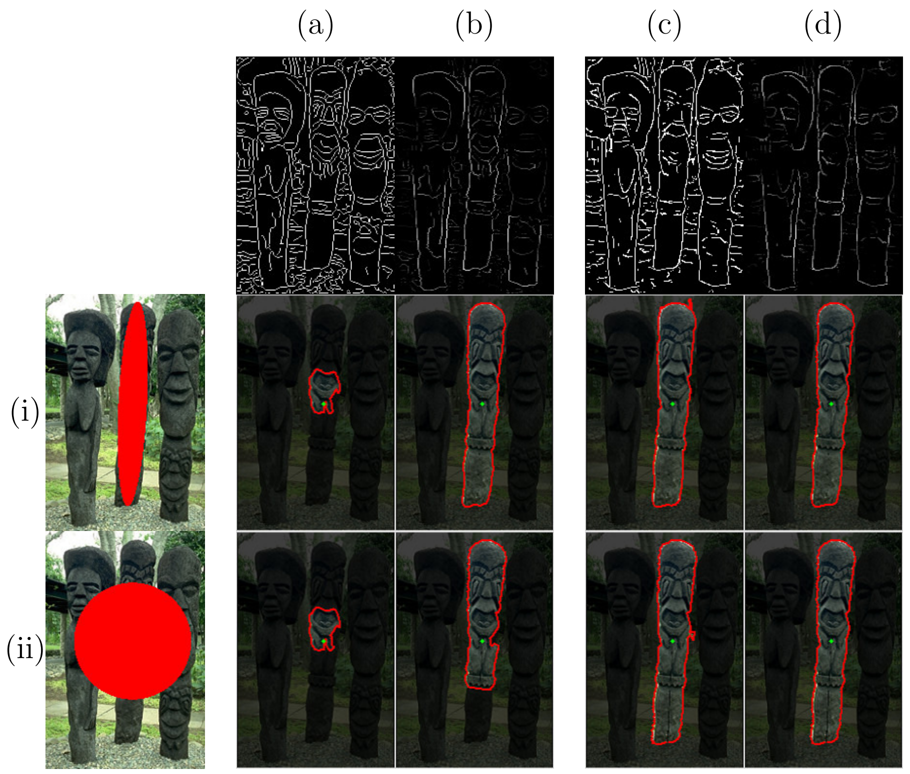


Figure 3.38: Segmentation using Elliptic Polar Coordinate: This figure shows a matrix top raw shows edges, and left column shows shape of ellipse used for segmentations. (a) and (b) are Canny edge map and strengthened Canny edge map respectively. (c) and (d) are Berkeley edge map and strengthened Berkeley edge map respectively. ellipse adjusted for an object in the image is used for (i). For comparison, standard polar coordinate is used for (ii).

minor axis respectively. The transformation between (x, y) and (r, θ) is defined in eq. (3.39). We applied this method to an image in the Berkeley image dataset. An ellipse close to an object shape was given manually. It is verified that the similar segmentations to the ones in the previous experiment can be obtained by adjusting the weight in the cost function as explained above without actual image warping to polar coordinate system.

3.6 Application to Motion

Up to this point, the torque operator was applied to edges in appearance. However, the torque operator is applicable to geometric edges, such as depth edges and motion edges. It is expected that the torque operator applied to these geometric edges will perform better for finding objects. We demonstrate here to apply the torque operator to motion edges. Torque value maps and figure-ground segmentations based on torque using flow are demonstrated. The torque value map was computed for each frame using norm of optical flow. Motion boundaries were explicitly detected, and torque maps were computed by the way of gradient-based torque computation. Segmentation was simply done with separation of positive and negative torque value regions. Fig. 3.39 and 3.40 show test sequence, optical flow, torque value map, and foreground segmentation. First row shows every ten frames of a test image sequence. Second row shows color coded optical flow maps. Third row shows torque value maps based on optical flow. Fourth row shows segmentations using the torque value maps. Since the method of the segmentation here was simple, there was a lot of room for improvement It was verified that torque maps were well representing regions of motion even though background was moving.

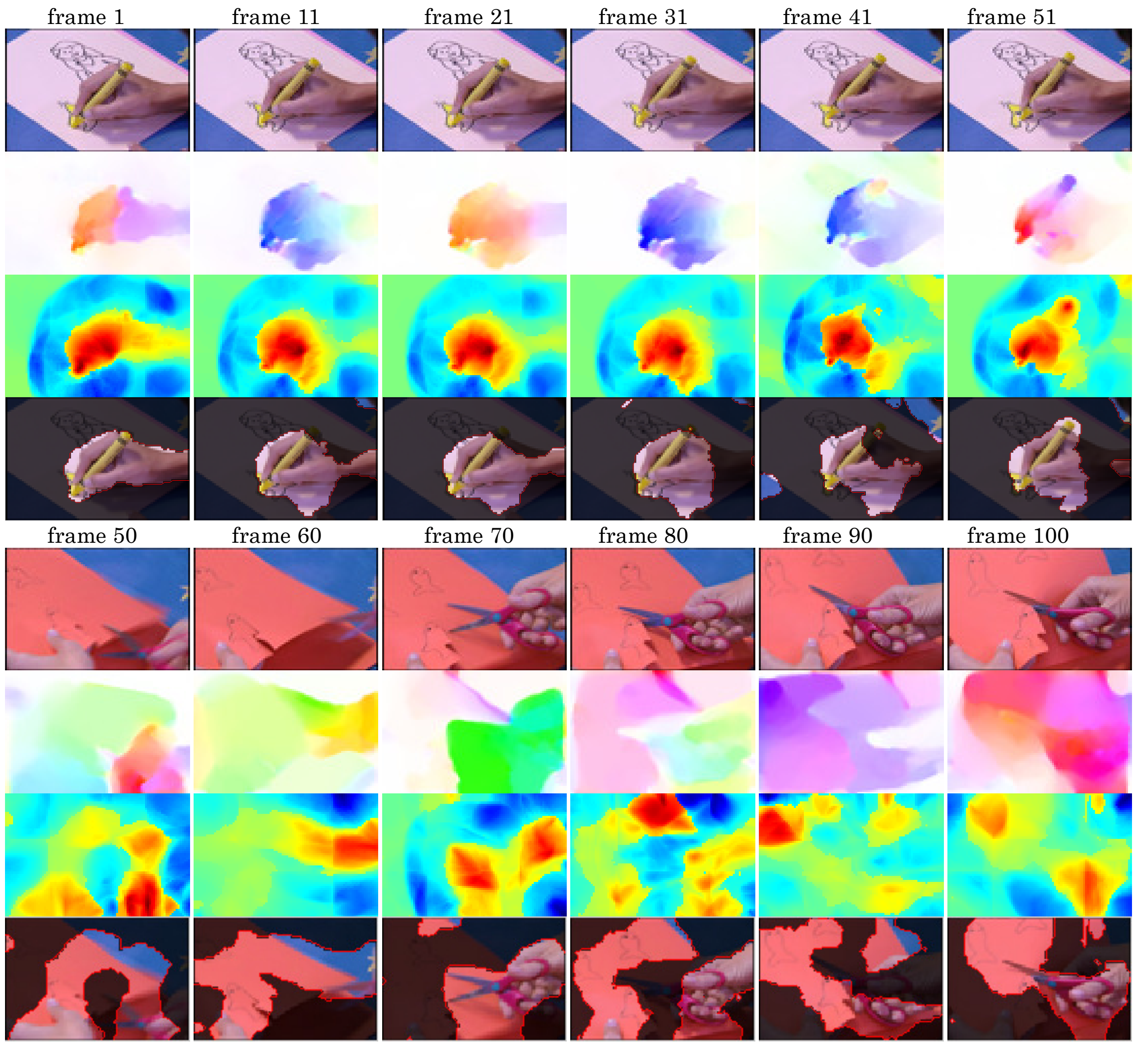


Figure 3.39: Motion-based Torque Value Maps and Segmentations 1/2: First row shows every ten frames of a test image sequence. Second row shows color coded optical flow maps. Third row shows torque value maps based on optical flow. Fourth row shows segmentations using the torque value maps.

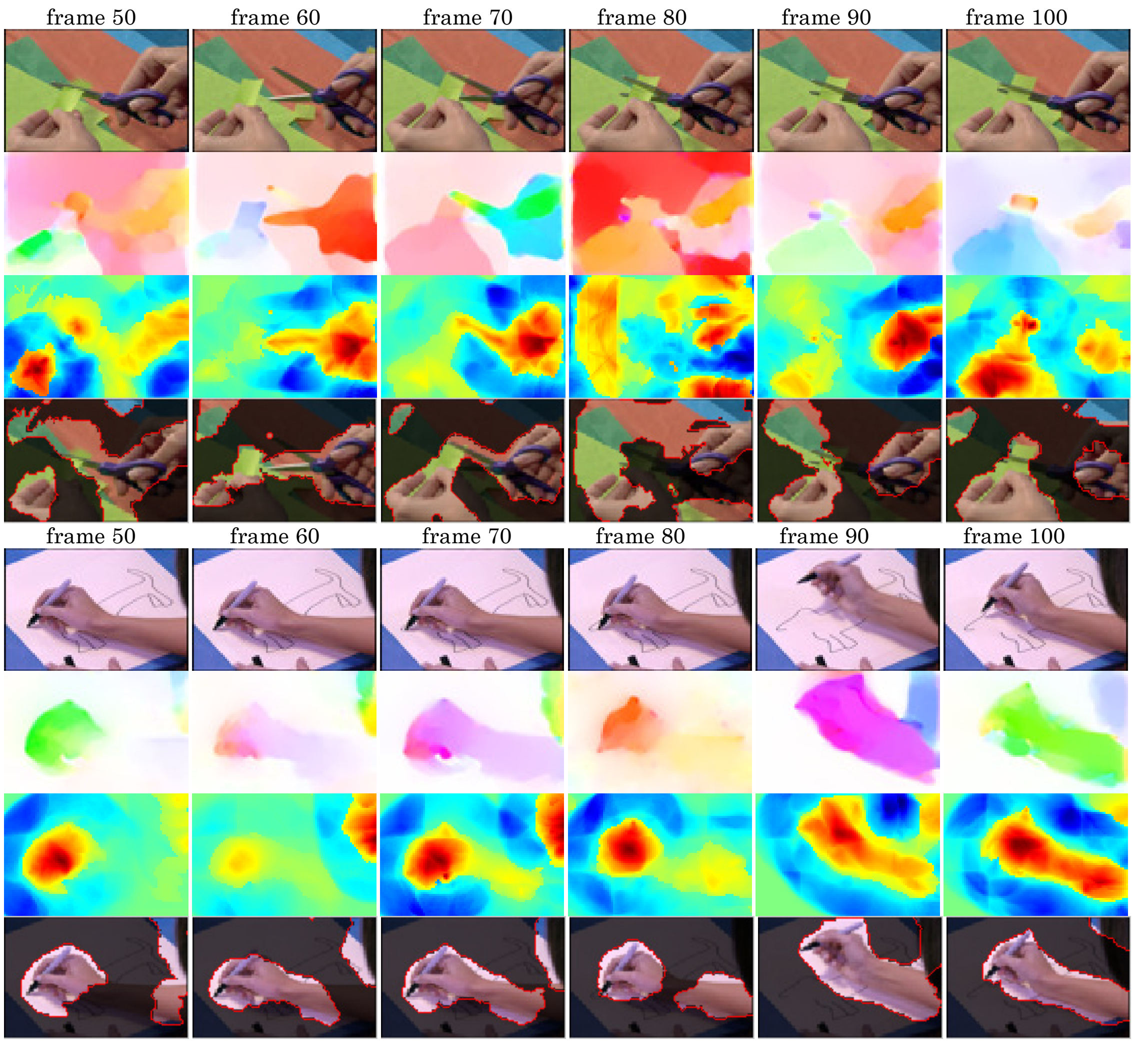


Figure 3.40: Motion-based Torque Value Maps and Segmentations 2/2: First row shows every ten frames of a test image sequence. Second row shows color coded optical flow maps. Third row shows torque value maps based on optical flow. Fourth row shows segmentations using the torque value maps.

Chapter 4

Conclusion

We introduced the "Torque Operator", a new mid-level visual operator, by giving the concept and proper definitions. We defined the torque for theoretical continuous and for practical discrete edge points and a torque based on image gradients. Fundamental properties, such as the relationship between torque and area, the usefulness of torque extrema, and torque value maps for simple shapes were explored using both theory and experiments. It was shown that the absolute value of the torque for a patch without normalization is equivalent to twice of the area enclosed by the edge segments and the lines connecting the center to the end points of the edge segments. For the torque based on image gradient, the absolute value of the torque for a patch is equivalent to the difference between the average intensity inside the patch and the average intensity on the patch boundary. A basic property of the torque operator is that it tends to generate larger values when edges are aligned in a way surrounding the center of the patch, and its scale matches the size of the patch. Therefore, torque extrema are expected to indicate the center of surrounding edges, which is often the center of an object. The benefit of these characteristic properties was demonstrated in three applications: attention, boundary detection, and segmentation. For all these applications, it was shown quantitatively, that the torque operator enhances the performance of existing techniques. Furthermore, the potential of torque operator when applied to geometric cues, such as depth edges and motion edges, was demonstrated. For practical use, a fast computation and approximation of the torque computation were also introduced.

Being between local operators (like the derivative operators) and global operators (like multidimensional histograms of features), mid-level operators have the potential to support many computer vision processes. Important key properties of the torque operator were discussed and verified by experiments, but we might have scratched only the surface of a new mid-level operator in computer vision. In the experiments of this thesis, we have not yet utilized the benefit of torque operator in the most efficient way. Only strengthened edges were used in the experiments on segmentation, but the torque value itself is also expected to be a useful visual cue for segmentation. Furthermore, since the torque operator indicates the scale of the surrounding edge structure, *e.g.* object-hood, it could be used to determine the region of attention.

In this thesis, we applied the torque operator only to square and circular patches. However, further extension of this work could be using patches of other shapes such as rectangular and elliptic patches so that the torque operator could be adapted to elongated objects. Another direction of extension is applying the torque operator on different cues, such as for example depth.

We showed benefits of the torque operator for a variety of vision tasks through qualitative and quantitative evidence. There could be other benefits for many methods by utilizing features extracted by the torque operator. It remains for future research.

Appendix A

Segmentation for Objects with Shadow

A.1 Algorithm

Object segmentation in images based on color or intensity edges is often disturbed by edges of shading and shadow. The shadow could be casted onto background and object itself due to the shape of the object. Even though the brightness changes by the shading and self-casted shadow, it is expected that the hue stays similar. Therefore, the first step is to weaken the edges at which magnitude of hue gradient is small, which is formulated as follows:

$$Pb^{hue}(p) = \begin{cases} Pb(p) \cdot \alpha & \left|g^{hue}(p)\right| < c_0, \\ Pb(p) & \text{otherwise}, \end{cases} \tag{A.1}$$

where $Pb(p)$ is the boundary probability at a point p. g^{hue} is gradient in hue channel. $\alpha < 1$ is a constant factor to weaken the boundary probability. Since hue is represented by value in ring, such as angle between $-\pi$ and π, the gradient of hue have to be computed by taking this circulation of value into account. Fig. A.1 shows an example to show that edges due to shading are weakened by this method, and a better segmentation is obtained. We assume in the explanation and demonstration a fixation-based segmentation as the segmentation method where a fixation point is given.

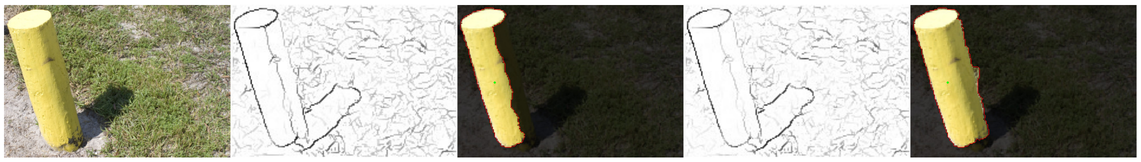

(a) (b) (c) (d) (e)

Figure A.1: Segmentation using Hue Gradient. (a) test image. (b) edge map by Pb[15]. (c) segmentation using Pb. (d) edge map computed by eq. (A.1). (e) segmentation using the edge map in (d). The green dot in a segmentation is the given fixation point.

Although effects of edges due to shading and self-casted shadow is reduced by using hue channel, failure of segmentation could be caused by the shadow casted on the background. We even observed that the edge weakening using the hue gradient sometimes weaken the edge between object and the shadow casted on the background, and cause a segmentation including the shadow as a part of object. Fig. A.2 is an example of the problematic segmentation due to shadow casted onto the background. As taking a close look at the boundary between the hydrant and the shadow in the edge map in the figure, it is noticed that there is no clear object boundary edge due to similarity of color between the shaded part of object and the shadow, and this missing of an edge causes the wrong segmentation.

The shadow detection is expected to improve segmentation quality by solving the issue on wrong segmentation due to shadows. Once the shadow casted onto the background is detected, one can strengthen the boundary of the casted shadow inside the intermediate segmentation, and weaken the boundary of the casted shadow coinciding with the intermediate segmentation boundary. This strategy is depicted

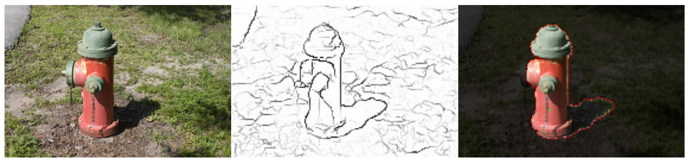

(a) (b) (c)

Figure A.2: Example of Problematic Segmentation due to Shadow. (a) test image. (b) edge map by combining Pb and hue gradient. (c) segmentation using edge map in (b).

in the Fig. A.3. In this figure, for the sake of simplicity, we assume that we obtain edges of the image in (a) as in (b) using hue gradient check explained above, and its intermediate segmentation using edge map of (b) is just the area enclosed by the edges, which corresponds to the segmentation in Fig. A.2(c) for example. If we could extract the shadow region casted onto the background in red in (c), the boundary of the shadow region can be computed. Then, the part of boundary inside the intermediate segmentation is added as an edge (drawn in green), and the other part of boundary coinciding with the intermediate segmentation boundary is weaken (drawn in blue). Then, the new boundary probability map is computed as follows:

$$Pb^{sdw}(p) = \begin{cases} \max\left(Pb^{hue}(p) + \beta^{+}, 1\right) & p \in \partial R^{sdw} \cap R^{seg}, \\ \max\left(Pb^{hue}(p) - \beta^{-}, 0\right) & p \in \partial R^{sdw} \cap \partial R^{seg}, \\ Pb^{hue}(p) & \text{otherwise}, \end{cases} \tag{A.2}$$

where R^{sdw} and R^{seg} are shadow region and intermediately segmented region respectively. ∂R denotes the boundary of region R. β^{+} and β^{-} are positive constant parameters to strengthen and weaken edges respectively. Here, Pb^{hue} is strengthened by torque operator based on edges extracted by hue gradient. The shadow region R^{seg} is extracted from shadow probability map by binalizing the map by a threshold because usually the shadow casted on the background has higher probability value than the probability value for self-casted shadow and shading. Connected components are computed after the binalization to determine the boundaries of shadow regions.

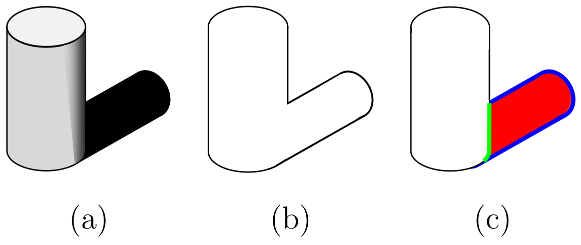
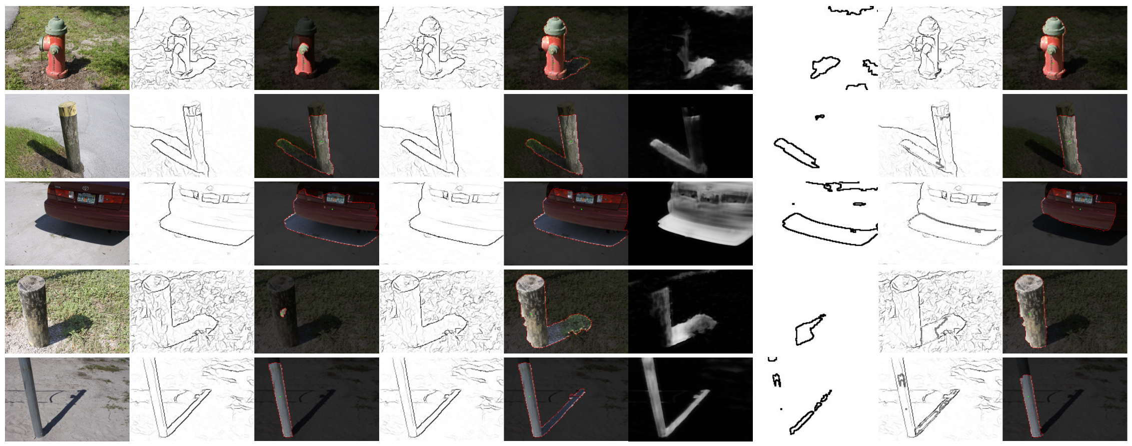


Figure A.3: Strategy of Edge Strengthening and Weakening. (a) A object with shading and shadow. (b) Detected edges. It corresponds to intermediate segmentation. (c) Casted shadow in red is detected. Boundary of the shadow in side the segmentation in green is strengthened (added) and boundary of shadow coinciding with the intermediate segmentation boundary in red is weakened.

A.2 Experiment

We demonstrate the segmentation improvement by this method. Fig. A.4 visualizes resulting improvements of segmentations. In the figure, it can be seen that segmentation is disturbed by shading and self-casted shadow in the segmentation for the first and fourth test images in (c). This problem is overcome using hue gradient, and it results in segmentations in (e). The fifth test image is the case that the segmentation in (c) becomes worse in (e) including shadow as a part of the object by edge weakening using hue gradient, but this is treated in the next step. At the intermediate segmentation shown in (e), they are all suffered from the shadow casted onto background. This problem is overcome by using shadow detection. The shadow regions are extracted by binalizing the the shadow probability map in (f) with a threshold, and boundaries of connected components of the shadow regions are shown in (g). At the final segmentations in (g), they are improved comparing to the segmentation in (c) and (e). Edge maps computed by eq. (A.2) shown in (h) are then used for segmentation shown in (i). In this demonstration, the following parameter set is used: $c_0 = 0.01$, $\alpha = 0.5$, $\beta^+ = 0.5$, $\beta^- = 0.5$ for all test images. The threshold to extract shadow region were 0.3 for test images in the first and second row, and 0.6 for the rest of test images.



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

Figure A.4: Object Segmentation for Images with Shadow. (a) Test image. (b) Edge map by Pb[15]. (c) Segmentation using Pb. (d) Edge map computed by eq. (A.1). (e) Segmentation using the edge map strengthened by torque operator based on edges in (d). This segmentation is used as intermediate segmentation in eq. (A.2). (f) Shadow probability map. (g) Boundaries of extracted shadow region casted onto background. (h) Edge map computed by eq. (A.2). (i) Segmentation using the edge map in (h).

Bibliography

[1] David Marr. *Vision: a computational investigation into the human representation and processing of visual information*. W. H. Freeman and Company, San Francisco, CA, 1982.

[2] Abhijit S. Ogale and Yiannis Aloimonos. A roadmap to the integration of early visual modules. *IJCV Special Issue on Early Cognitive Vision*, 72(1):9–25, April 2007.

[3] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer, Boston, 1994.

[4] P. Parent and S.W. Zucker. Trace inference, curvature consistency, and curve detection. *PAMI*, 11(8):823–839, 1989.

[5] E.M. Arkin, L.P. Chew, D.P. Huttenlocher, K. Kedem, and J.S.B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *PAMI*, 13(3):209–216, 1991.

[6] J. Zunic and P.L. Rosin. A new convexity measure for polygons. *PAMI*, 26(7):923–934, 2004.

[7] Padraig Corcoran, Peter Mooney, and Adam Winstanley. A convexity measure for open and closed contours. In *BMVC*, 2011.

[8] Xiaofeng Ren, Charless C. Fowlkes, and Jitendra Malik. Cue integration in figure/ground labeling. In *Advances in Neural Information Processing Systems*, 2005.

[9] S. Zheng, A. Yuille, and Z. Tu. Detecting object boundaries using low-, mid-, and high-level information. *CVIU*, 114(10):1055–1067, 2010.

[10] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, April 2002.

[11] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *CVPR*, 2004.

[12] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *BMVC*, 2004.

[13] A. Blake J. Shotton and R. Cipolla. Contour-based learning for object detection. In *ICCV*, pages 503–510, 2005.

[14] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *ECCV*, pages 575–588, 2006.

[15] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, 2004.

[16] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *CVPR*, pages 1–8, 2008.

[17] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011.

[18] I. Leichter and M. Lindenbaum. Boundary ownership by lifting to 2.1d. In *ICCV*, 2009.

[19] A. Hollingworth, C.C. Williams, and J.M. Henderson. To see and remember: Visually specific information is retained in memory from previously attended objects in naturalscenes. *Psychonomic Bulletin and Review*, 8:761–768, 2001.

[20] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *PAMI*, 20(11):1254–1259, 1998.

[21] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *NIPS*, pages 545–552, 2006.

[22] Wolfgang Einhäuser, Wolfgang Kruse, Klaus-Peter P. Hoffmann, and Peter König. Differences of monkey and human overt attention under natural conditions. *Vision research*, 46(8-9):1194–1209, apr 2006.

[23] L. Holm, J. Eriksson, and L. Andersson. Looking as if you know: Systematic object inspection precedes object recognition. *Journal of Vision*, 8(4):1–7, 2008.

[24] W. Einhäuser, M. Spain, and P. Perona. Objects predict fixations better than early saliency. *Journal of Vision*, 8(14):1–26, 2008.

[25] Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *CVPR*, pages 1–8. IEEE Computer Society, 2007.

[26] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[27] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *CVPR*, pages 53–60, Washington, DC, USA, 2006. IEEE Computer Society.

[28] Pei Yin, Antonio Criminisi, John Winn, and Irfan Essa. Tree-based classifiers for bilayer video segmentation. In *CVPR*. IEEE Computer Society, 2007.

[29] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bilayer segmentation of binocular stereo video. In *CVPR*. IEEE Computer Society, 2005.

[30] Woontack Woo, Namgyu Kim, and Yuichi Iwadate. Object segmentation for z-keying using stereo images. In *ICSP*, 2000.

[31] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, May 2002.

[32] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[33] C.V. Rother and V. Kolmogorov. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[34] Andrew Blake and Michael Isard. *Active Contours*. Springer, 2000.

[35] D. Mumford and J. Shah. Functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[36] S. J. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed; algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[37] Ajay Mishra, Yiannis Aloimonos, and Cheong Loong Fah. Active segmentation with fixation. In *ICCV*, pages 468–475, 2009.

[38] Luminita A. Vese and Tony F. Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *IJCV*, 50(3):271–293, 2002.

[39] Piotr Dollár, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *CVPR*, New York City, 2006.

[40] Xiaofeng Ren. Multi-scale improves boundary detection in natural images. In *ECCV*, 2008.

[41] M. Kampel, A. Hanbury, P. Blauensteiner, and H. Wildenauer. Improved motion segmentation based on shadow detection. *ELCVIA*, 6(3):1–12, December 2007.

[42] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, volume 1, pages 10–17, 2003.

[43] Pushmeet Kohli, L'Ubor Ladický, and Philip H. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009.

[44] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

[45] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006.

[46] T. Schoenemann, F. Kahl, and D. Cremers. Curvature regularity for region-based image segmentation and inpainting: A linear programming relaxation. In *ICCV*, Kyoto, Japan, 2009.

[47] Dorit S. Hochbaum and Vikas Singh. An efficient algorithm for co-segmentation. In *ICCV*, Kyoto, Japan, 2009.

[48] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In *CVPR*, 2006.

[49] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *ICCV*, Kyoto, Japan, 2009.

[50] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *ICCV*. IEEE Computer Society, IEEE, Oct. 2007.

[51] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[52] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *ECCV*, volume 3024 of *LNCS*, pages 25–36, Prague, Czech Republic, May 2004. Springer-Verlag.

[53] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers. Duality tv-l1 flow with fundamental matrix prior. In *IVCNZ*, pages 1–6, 2008.

[54] Henning Zimmer, Andrés Bruhn, Joachim Weickert, Levi Valgaerts, Agustín Salgado, Bodo Rosenhahn, and Hans-Peter Seidel. Complementary optic flow. In *EMMCVPR*, pages 207–220, Berlin, Heidelberg, 2009. Springer-Verlag.

[55] J. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *IJCAI*, 2001.

[56] Talya Meltzer, Chen Yanover, and Yair Weiss. Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation. In *ICCV*, volume 1, pages 428–435, Washington, DC, USA, 2005. IEEE Computer Society.

[57] D. Marr and T.A. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, October 15, 1976, October 1976.

[58] C. Lawrence Zitnick and Takeo Kanade. A cooperative algorithm for stereo matching and occlusion detection. *PAMI*, 22(7):675–684, 2000.

[59] Roland Brockers. Cooperative stereo matching with color-based adaptive local support. In *CAIP*, pages 1019–1027, Berlin, Heidelberg, 2009. Springer-Verlag.

[60] Y. Zhang and C. Kambhamettu. Stereo matching with segmentation-based cooperation. In *ECCV*, pages II: 556–571. Springer-Verlag, 2002.

[61] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769. IEEE Computer Society, 2004.

[62] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[63] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423, 2001.

[64] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[65] T. Judd, K. Ehinger, F. Durand, and A. Torralba. Learning to predict where humans look. In *ICCV*, pages 2106–2113, 2009.

[66] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *PAMI*, 28:594–611, April 2006.

[67] John Canny. A computational approach to edge detection. *PAMI*, 8(6):679–698, 1986.

[68] A. N. Stein, T. S. Stepleton, and M. Hebert. Towards unsupervised whole-object segmentation: Combining automated matting with boundary detection. In *CVPR*, pages 1–8, 2008.

[69] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, pages 2294–2301, 2009.