Pattern Recognition 44 (2011) 2008–2016

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Interactive segmentation of non-star-shaped contours by dynamic programming

Xiaoyi Jiang *, Andree Große, Kai Rothaus

Department of Mathematics and Computer Science, University of Münster, Einsteinstrasse 62, D-48149 Münster, Germany

ARTICLE INFO

Available online 22 March 2011

Keywords:
Contour detection
Non-convex
Non-star-shaped
Shortest path
Dynamic programming

ABSTRACT

In this paper we present the RACK algorithm for the detection of optimal non-star-shaped contours in images. It is based on the combination of a user-driven image transformation and dynamic programming. The fundamental idea is to interactively specify and edit the general shape of the desired object by using a rack. This rack is used to model the image as a directed acyclic weighted graph that contains a path corresponding to the expected contour. In this graph, the shortest path with respect to an adequate cost function can be calculated efficiently via dynamic programming. The experimental results indicate the algorithm's ability of combining an acceptable amount of user interaction with generally good segmentation results.

1. Introduction

Dynamic programming (DP) is a popular technique for contour segmentation due to its elegance and guarantee of optimality [1–3]. However, the contours that can be handled by such techniques are limited to star-shaped only. Some approaches have been proposed to deal with more complex shapes [3–5] by means of increased human interaction or additional prior information. In this work we address the task with a new approach that will be called the RACK algorithm. The idea is to specify and edit the general shape of the desired object by using a rack; see Fig. 7. This rack is used to model the image as a directed acyclic graph (DAG) that contains a path corresponding to the expected contour. With the image's edge strength or whatever other suitable measures as graph weights, an optimal contour can be defined as the shortest path in the DAG with respect to some cost function, which can be globally minimized via dynamic programming.

Our approach is in line with interactive image segmentation. Fully automated segmentation is known to be an ill-posed problem due to the fact that there is neither a clear definition nor an objective goodness measure of a semantic segment. The user intention is manifold and depends on applications. In order to perform a semantically meaningful segmentation, it is thus helpful to take *a priori* information about the objects into account. Interactive segmentation algorithms provide a solution by invoking the aid of a human operator to supply the high-level information needed to detect and extract semantic objects through a series of interactions. For instance, the user may mark (small) areas of the image as object or background and the algorithm updates the segmentation using the new information. By iteratively providing more interactions the user can refine the segmentation. The goal of interactive segmentation is thus to provide a means of extracting semantic objects from an image quickly and accurately.

The early approach of seeded region growing [6] is a simple and computationally inexpensive technique for interactive segmentation based on a set of seed points. The more sophisticated approach from [7] uses the seed pixels to estimate the labels of unlabeled pixels by learning on probabilistic hypergraphs. Intelligent scissors [8] allows a user to choose a minimum cost contour by roughly tracing the object's boundary with the mouse. Active contours or snakes [9] evolutionally improve a manually specified initial contour. The interactive graph cut algorithm [10] formulates the interactive segmentation problem within a MAP-MRF framework, subsequently determining a globally optimal solution using a fast mincut/max-flow algorithm. Two further variants of this category are the GrabCut algorithm [11] and the lazy snapping algorithm [12]. The work [13] extends the classical region growing to a maximal similarity based interactive segmentation. A linear programming approach is presented in [14]. Recently, a comparative evaluation of four interactive segmentation algorithms is reported in [15]. While these works are devoted to general object extraction from photographs and natural scenes, there are also application-specific demands and related developments. For instance, interactive segmentation methods are indispensable for biomedical image analysis [16,17].

The remainder of this work is organized as follows. We discuss related works on DP-based contour detection in Section 2. In Section 3, we describe our algorithm according to the three steps shown in Fig. 1. The image is first modeled as an undirected graph and then turned into a DAG. As a third step, the shortest path is calculated in this DAG, which represents an optimal contour. The key issue of transforming the undirected graph into a DAG guided by a user-specified rack is tackled in Section 4. Experimental

* Corresponding author.
E-mail address: xjiang@uni-muenster.de (X. Jiang).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.03.010

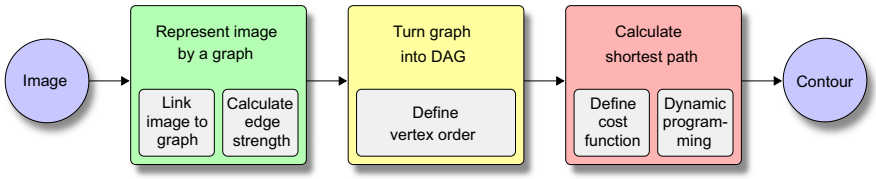

Fig. 1. Outline of the Rack algorithm.

results are shown in Section 5. Finally, we conclude with a discussion in Section 6. This paper is an extended version of the conference version [18] and contains more algorithmic details and experimental validation.

2. Dynamic programming for contour detection

The principle of dynamic programming was introduced by Bellman [19]. Dynamic programming is a very efficient way of finding an optimal path in a matrix of weights with an unknown start point in the first column and an unknown end point in the last column [20–22]. Among all possible paths from left to right the optimal path maximizes (or minimizes) the sum of weights of all elements on the path. If the matrix represents the edge magnitude, then the optimal path found by the dynamic programming simply means the best contour from left to right, which maximizes the sum of edge magnitudes. Let S denote the weight matrix. The optimal path is computed by means of a cumulative weight matrix C in a column-row manner. The first column of C is initialized by the corresponding values from S. Starting from the second column, the cumulative weight at position (i,j) is computed for each column j as

$$C(i,j) = S(i,j) + \max_{n \in N(i,j)} [C(n) + T(n,(i,j))]$$

where $N(i,j)$ is the set of possible predecessors of (i,j) in the previous column $j-1$. In the case of 8-neighborhood, $N(i,j)$ is usually set to be $\{(i-1,j-1), (i,j-1), (i+1,j-1)\}$. $T(n, (i,j))$ represents the transition weight from the predecessor n to the current position (i,j). In addition to computing $C(i,j)$, a pointer is set to the predecessor n that achieves the maximum among all predecessors in $N(i,j)$. To determine the optimal path we follow the pointers from the last column backwards to the first column, starting from the position in the last column of C with the maximal value.

The standard dynamic programming can be easily extended to handle closed contours. The fundamental idea is to select a point p in the interior of the contour. Then, a polar transformation with p being the central point brings the original image into a matrix, in which a closed contour becomes a contour from left to right. The optimal path computed in this polar space, however, does not guarantee a closed contour in the original image space. Several exact and approximate approaches have been proposed [23,24] to enforce a circular solution. The principle of DP-based contour detection has also been extended to deal with simultaneous finding of multiple contours [1,2].

The DP-based technique has several advantages that make it attractive for object detection. First, the calculation of an optimal contour can be efficiently implemented. Furthermore, the calculated contour will represent a global optimum. This is a major difference to algorithms like active contours and makes the algorithm quite stable against local perturbations or gaps in the contour.

The main limitation of DP-based approaches is the restriction to star-shaped contours including convex contours as a special case only. A star-shaped contour is characterized by the existence of a point p such that for each point q of the contour the segment pq lies entirely within the contour. The set of all points p with the described property is called the contour's kernel. Theoretically, any point from the kernel may be used as the origin of the polar transformation. But practically, more centered kernel points tend to provide better detection results.

Several suggestions have been made toward dealing with shapes that are more complex than star-shaped. The segmentation tool Corridor Scissors [25] allows the user to mark a coarse circular corridor area around the object. Then, the algorithm searches for a shortest circular path inside the corridor. Two rather heuristic techniques are proposed in [3]. In one of them, a dual band of elliptic shape needs to be carefully placed so that the donut shape within the band can be transformed into a rectangular image in the polar space. In [4] a basic contour is first marked manually. Normals to this contour are then constructed. The length of the normals restricts the search space. The pixels along the normals are then transferred to polar coordinate system. However, the problem with this method is that it can only be applied to objects that are somewhat convex in nature. At places where the object has sharp corners or pointed peaks, the normals will cross each other and thus the contour finding may fail. In order to solve this problem, the method from [5] restricts the search space to a dual band of predefined width. The inner and outer boundaries of the search space are interlinked with optimally determined straight lines that are guaranteed not to cross each other and thus allow to deal with complex shapes that cannot be processed by considering the normals only.

3. Rack-based contour detection by dynamic programming

Our approach substantially differs from the previous ones. In our case the user is required to interactively specify a rack (a kind of skeleton [26]) of the expected shape. Then, we adaptively construct a DAG based on the rack and solve the contour detection problem by calculating the shortest path in this graph by means of dynamic programming. An outline of our Rackalgorithm can be found in Fig. 1. We start with the fundamental graph representation of an image (Section 3.1). Then, the adaptive DAG construction is described in Section 3.2. We present how to calculate the shortest path in the constructed DAG in Section 3.3. Finally, the computational complexity of the rack-based contour detection algorithm is discussed in Section 3.4.

3.1. Graph representation of an image

Fig. 2 illustrates how an image of size $M \times N$ is modeled by a grid graph of size $(M+1) \times (N+1)$. Each vertex represents a boundary point between adjacent pixels. A contour in the image is expressed by a path in this graph.

A weight w is assigned to each graph edge indicating the image's edge strength at the corresponding position in the image or whatever measure useful for contour detection. We assume that the weight is normalized, i.e. $w : E \rightarrow [0,1]$. Fig. 2(c) illustrates the weight for each edge as grayscale highlights, where black indicates high edge strength. This grid graph therefore is a weighted undirected graph $G=(V,E,w)$. We assume that a weight of 0 corresponds