Lưu đồ thuật toán (Algorithm Diagram):

**1. Edge Enhanced MSER**

Sự tương phản của các đối tượng chữ cái với nền và sự đồng nhất về màu sắc giữa chúng chính là yếu tố quan trọng trong việc tìm kiếm sự hiện diện của chữ cái. Ở đây, chúng tôi dùng MSER (maximally stable extremal regions) để nhận diện. Nhưng thuật toán này không nhạy với việc phát hiện chữ cái trên những hình ảnh có độ phân giải kém. Vì vậy, chúng tôi sử dụng MSER kết hợp với Canny edges.

Chuyển hình ảnh ban đầu về dạng trắng đen, sau đó sử dụng thuật toán phát hiện cạnh Canny edges để phát hiện vùng bao của vùng cực trị (extremal region1).

**1 Hình ảnh** I là một ánh xạ I : D \subset \mathbb{Z}^2 \to S.

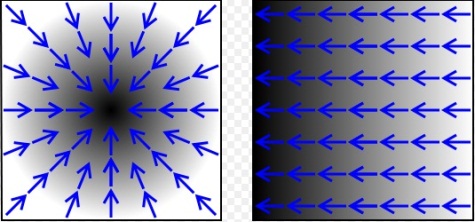
S là tập các điểm trong D thỏa quan hệ phản xứng, bắt cầu, hoàn toàn.

**Tập** Q \subset D sẽ là vùng cực trị nếu một trong hai trường hợp sau xảy ra:

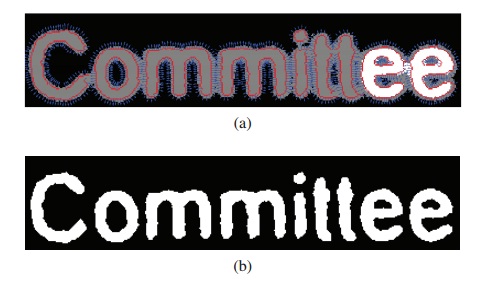
-Tất cả các điểm  p \in Q, q \in \partial Q : I(p) > I(q) thì ta gọi Q là vùng cường độ cực đại.

-Tất cả các điểm  p \in Q, q \in \partial Q : I(p) < I(q) thì ta gọi Q là vùng cường độ cực tiểu.

Loại bỏ tất cả các điểm ảnh MSER nằm ngoài vùng bao của Canny edges theo hướng dốc (hướng dốc là hướng theo hướng của blue-arrows 2). Hướng dốc (hướng của blue-arrow) trong hình a hướng từ vùng chữ (màu sáng) ra vùng nền màu tối.

2 Blue-arrows là khái niệm thường được dùng trong thị giác máy tính, dùng để chỉ hướng thay đổi cường độ sáng hoặc màu sắc trong một bức ảnh. Đối với ảnh trắng đen, mũi tên có hướng từ vùng có màu sáng vào vùng có màu tối, người ta còn gọi là hướng dốc.

Ta có kết quả như sau:



**Hình 1**

a) Thuật toán Canny edges phát hiện các cạnh (màu đỏ) bao quanh vùng cực trị.

b) Hình ảnh sau khi đã loại bỏ các điểm ảnh MSER nằm ngoài vùng bao theo hướng dốc của blue-arrows.

Cách làm này không chỉ tăng hiệu năng của Geometric Filtering mà còn giúp MSER hoạt động tốt trong những điều kiện ảnh mờ khác nhau.

**2. Geometric Filtering**

Ở bước này, chúng ta sẽ xác định và loại bỏ những đối tượng không phải là chữ.

Sau bước 1 ta có thể thu về một ảnh nhị phân (binary image3).

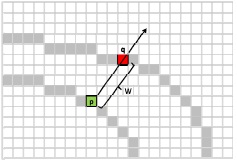
3 Ảnh nhị phân là hình ảnh mà mỗi điểm ảnh chỉ bao gồm 2 giá trị, có nghĩa là bức ảnh 2 màu.

Hầu hết các chữ cái đều có tỉ lệ (aspect ratio4) xấp xỉ 1, ta sẽ loại bỏ tất cả các đối tượng có tỉ lệ rất lớn hoặc rất bé. Đồng thời ta cũng lựa chọn một ngưỡng tỉ lệ để đảm bảo rằng các chữ cái có dạng thon dài như “l” hay “i” sẽ không bị loại bỏ. Đồng thời ta cũng sẽ loại bỏ các đối tượng có nhiều lỗ, vì đó không phải là ứng cử viên phù hợp cho chữ cái.

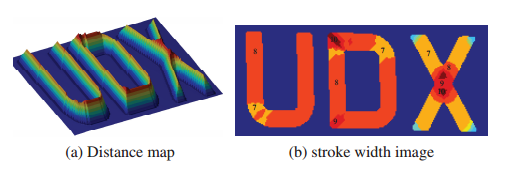
4 Aspect ratio của một chữ cái là tỉ lệ giữ chiều dài và chiều rộng của chữ cái đó, thông thường các chữ cái thường có tỉ lệ chiều dài và chiều rộng xấp xỉ nhau, tuy nhiên có trường hợp ngoại lệ là “l” và “i” có tỉ lệ lớn, ta sẽ chọn ngưỡng riêng cho 2 đối tượng này.

**3. Finding Stroke Width by Distance Transform**

Ở bước này ta sẽ tiến hành tìm độ dày (stroke width5) của các nét chữ. Việc tìm độ dày của nét chữ là một trong những bước rất quan trọng trong việc nhận dạng chữ cái.

5Stroke width: Độ dày của nét chữ ở điểm p chính là w - khoảng cách từ p đến q.

Ở đây, chúng tôi gán nhãn độ dày cho các điểm ảnh nằm bên trong vùng được bao bởi thuật toán Canny edges (được xác định ở bước 1). Bắt đầu với những điểm ảnh nằm ở chính giữa nét chữ, ta sẽ gán giá trị độ dày cho điểm ảnh bằng khoảng cách Euclide của nó đến điểm ảnh gần nhất nằm trên cạnh Canny (những điểm ảnh nằm ở rìa). Cứ như vậy tính toán lan truyền từ những điểm nằm ở giữa đến những điểm rìa ta sẽ được ảnh như hình 2b. Màu càng đỏ đậm thì giá trị độ dày tại điểm đó càng lớn.



**Hình 2**

a) Ảnh khoảng cách

b) Ảnh độ dày

Mã giả của các thức tìm độ dày được mô tả dưới đây:

**Algorithm 1** Finding stroke width  
**Input:** binary image BW  
**Output:** stroke width image SW  
D := DistanceTransform(BW);  
D := round(D);  
**for** *p* = each foreground pixel in D **do**  
pVal := D(*p*);  
LookUp(*p*) := *p*’s 8 neighbors whose value *<* pVal;  
**end for**  
*{*LookUp can be efficiently computed without FOR loop.*}*  
MaxStroke := max(D);  
**for** Stroke = MaxStroke **to** 1 **do**  
StrokeIndex := find(D==Stroke);  
NeighborIndex = Lookup(StrokeIndex);  
**while** NeighborIndex not empty **do**  
D(NeighborIndex) := Stroke;  
NeighborIndex := Lookup(NeighborIndex);  
**end while**  
**end for**  
**return** SW := D;

**4. Text Line Formation and Word Separation**

Ở bước này, ta sẽ xác định định dạng dòng chữ và tách từ. Dòng văn bản là dấu hiệu quan trọng để xác định sự tồn tại của văn bản. Hầu hết các văn bản đều xuất hiện dạng đường thẳng hoặc đường cong nhẹ. Để xác định các dòng văn bản ta gom nhóm các đối tượng chữ cái thành từng cặp. Các chữ cái thuộc một dòng thường có độ dày và có cùng chiều gần tương đương nhau. Hai đối tượng chữ được ghép đôi khi và chỉ khi tỉ lệ về độ dày trung bình giữa hai chữ không vượt quá 1.5 và tỉ lệ về chiều cao giữa chúng không quá 2. Đồng thời nếu 2 chữ cái cách quá xa thì ta cũng không gom nhóm chúng thành một đôi.

Một dòng văn bản sẽ được hình thành khi ta kết nối các cặp đôi chữ cái lại với nhau. Một dòng có thể xác định bằng một đường thẳng đi qua nhiều trọng tâm của các cặp chữ nhất hoặc cũng có thể xác định bằng đường thẳng cắt nhiều cặp chữ nhất, lúc đó ta gom các cặp chữ nằm trên đường thẳng đó thành một cụm (cluster). Quá trình kết nối này được lặp lại cho đến khi tất cả các ứng cử viên chữ được gom cụm hoàn tất, hoặc trong trường hợp nếu ít hơn 3 đối tượng chữ cái trong cụm thì ta cũng sẽ dừng lại quá trình này. Nếu có từ 3 đối tượng chữ cái trở lên thì ta mới xem nó là một dòng văn bản.

Sau đó, trong những dòng văn bản được thuật toán đề xuất, ta sẽ phải tiếp tục sàn lọc, loại bỏ đi những dòng không phải là dòng văn bản thật sự như hình 3. Trong hình 3, ta thấy rằng thuật toán nhận diện dòng các cửa sổ máy bay cũng là một dòng văn bản (vì các cửa sổ có dạng hình chữ O nên nó cũng được nhận dạng là chữ cái từ đó suy ra nó cũng là dòng văn bản). Ta có thể tránh được điều này bằng các đối xứng các đối tượng chữ cái với nhau. Một dòng văn bản sẽ bị loại bỏ nếu các đối tượng chữ cái được lặp đi lặp lại. Hoặc dòng văn bản đó chứa số lượng lớn các đối tượng văn bản thân lồi (Như chữ O).

Cuối cùng, mỗi dòng văn bản được phân tách thành các từ riêng biệt. Để xác định, ta phải dựa trên khoảng trống giữa các kí tự, các từ. Có hai loại khoảng trống: khoảng trống giữa các chữ và khoảng trống giữa các từ. Chúng ta sẽ tính toán khoảng cách từ hình chiếu vuông góc của kí tự này đến kí tự kia dựa trên dòng văn bản và ta phân nó làm hai lớp dựa trên giải pháp của Otsu.