Trích xuất thông tin sự kiện từ tờ rơi

Yang Zhang*
Department of Geophyiscs
Stanford University
Stanford,CA
zyang03@stanford.edu

Hao Zhang*, HaoranLi*
Department of Electrical Engineering
Stanford University
Stanford, CA
{hzhang22, aimeeli}@stanford.edu

***Tóm tắt nội dung*—Trong học tập và nghiên cứu, chúng tôi đã phát triển một ứng dụng Android để lưu các thông tin về ngày và địa điểm của sự kiện trong các poster vào Google Lịch trên đám mây. Đầu tiên, một số phương pháp xử lý ảnh số đã được áp dụng để tiền xử lý những tấm ảnh poster, cái mà có thể được chụp từ điện thoại trong những trường hợp không tốt như điều kiện ánh sáng yếu, hoặc một phần bị che. Sau đó, chúng tôi cũng phải giải quyết phần background phức tạp của poster để dễ dàng hơn trong việc xác định phần chữ chứa thông tin quan trọng. Sau khi định vị được khu vực nội dung, phần chữ sẽ được chiết xuất bằng công cụ "Nhận dạng kí tự thị giác" (Optical Character Recognition - OCR) Tesseract. Việc xử lý này được hoàn thành ở một máy chủ và kết quả sẽ được trả về điện thoại. Cuối cùng, kĩ thuật "Xử lý ngôn ngữ tự nhiên" (Natural Language Processing - NLP) sẽ được tận dụng để phân tích các chuỗi kí tự, lọc ra được thời gian và địa điểm, rồi tự động lưu nó vào Google Lịch. Kết quả thực nghiệm cho thấy thuật toán của chúng tôi hoạt động tốt trong việc định vị khu vực chữ và trích xuất thông tin quan trọng từ poster.**

***Từ khóa*—** Android, Xử lý ảnh, OCR, trích xuất thông tin sự kiện

I. Giới thiệu

Mỗi ngày, chúng ta đối mặt với một lượng lớn các thông tin về sự kiện từ các tờ rơi dán trên tường, cửa ra vào, và thang máy trong các tòa nhà. Đó có thể là một buổi thuyết trình, một diễn đàn thảo luận hoặc một buổi hòa nhạc; tất cả đều kèm theo ngày, thời gian và địa điểm. Một số người có thể sẽ muốn lưu lại vào lịch những sự kiện thú vị khi họ đi ngang qua những poster đó. Tuy nhiên, vì những thông tin này không phải ở dạng số nên nó không thuận tiện cho chúng ta để lưu vào chiếc điện thoại thông minh. Do đó, chúng tôi hướng đến làm sao để xây dựng một hệ thống xử lý ảnh trên điện thoại, sử dụng phương pháp OCR, có khả năng tự động trích xuất thông tin sự kiện từ tờ rơi và sau đó trực tiếp tích hợp thông tin đó vào tài khoản Google Lịch của người dùng. Mục đích của chúng tôi là để thiết kế một công cụ trên nền tảng Android có thể cho phép người dùng ghi thông tin sự kiện (thời gian và địa điểm) vào tài khoản Google Lịch của họ chỉ đơn giản bằng cách chụp một tấm ảnh tờ rơi/poster như hình 1.

Các sinh viên khóa trước (EE368) đã tiến hành dự án "Thêm liên lạc tự động từ danh thiếp" [1]. Ở đây, chúng tôi chọn tờ rơi/poster, cái mà có nhiều biến thể cũng như các thành phần hình ảnh phức tạp. Do đó, công việc này sẽ mang tính thử thách cao hơn với yêu cầu nhận dạng chính xác chữ viết từ nhiều loại poster khác nhau. Chúng tôi cũng muốn cải tiến khả năng của hệ thống bằng cách cho phép người dùng chụp bức hình từ nhiều góc độ cũng như là độ rọi sáng khác nhau.

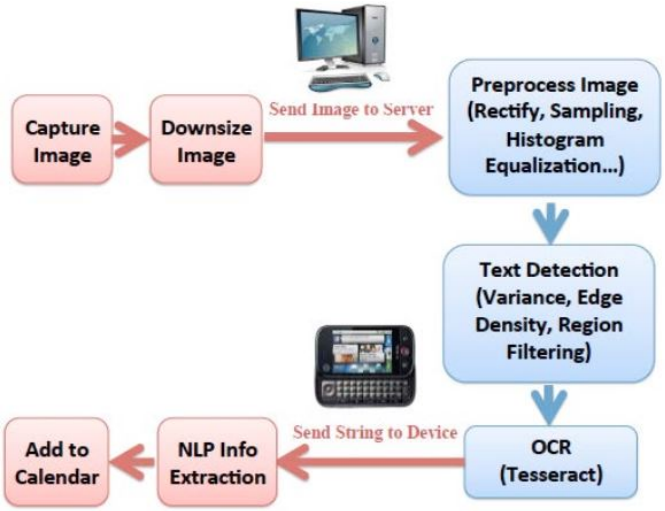
Hình 1. Dùng điện thoại thông minh chụp hình poster.

II. Tổng quan về hệ thống

Vì Tesseract OCR đòi hỏi xử lý phức tạp, nên khi vận hành phần mềm này trên điện thoại, tính đáp ứng thời gian thực của ứng dụng sẽ không khả thi. Do đó, chúng tôi quyết định đưa việc tiền xử lý và OCR lên một máy chủ có hiệu năng cao. Do đó, ứng dụng này trên Android sẽ sử dụng một framework cho hệ thống thông tin server-client [2]. Lưu đồ của hệ thống dạng đường ống được thể hiện ở hình 2. Các tầng thực thi chính của hệ thống:

1) Chụp và tải ảnh lên: Sử dụng thiết bị Android để chụp hình. Sau đó gửi ảnh lên máy chủ.

2) Tiền xử lý: Một vài tầng xử lý ảnh được thực hiện để chuẩn bị cho nhận dạng chữ bao gồm: điều chỉnh kích cỡ ảnh, nhận diện đường biên, cân bằng histogram, hiệu chỉnh tính đồng nhất. Mục III sẽ thảo luận chi tiết về việc này.



Hình 2. Sơ đồ hệ thống. Những khối đỏ được triển khai trên thiết bị Android. Những khối xanh thì nằm trên máy chủ.

3) Nhận dạng chữ: Đầu tiên, hai phương pháp thích ứng dựa trên sự thay đổi chữ và mật độ cạnh được sử dụng để định vị khu vực chứa chữ. Sau đó, các bộ lọc được đưa vào để làm giảm những vùng không có ích bằng diện tích, chiều cao, định hướng, độ liên kết. Điều này sẽ được bàn kĩ hơn trong mục III.

4) OCR: Những phần chứa chữ được gửi đến Tesseract OCR, còn kết quả sau đó sẽ được đóng gói thành một chuỗi kí tự và gửi ngược về điện thoại Android thông qua Internet. Mục IV sẽ thực hiện chuyện này.

5) Trích xuất thông tin NLP: Tập tin kí tự đã tải về sẽ được tách ra thành thời gian và địa điểm dựa trên một vài phương pháp xử lý ngôn ngữ tự nhiên. Công việc này sẽ được đề cập trong mục V.

6) Thêm sự kiện vào Google Lịch: Thông tin về ngày và địa điểm sẽ được thêm vào tài khoản Google Lịch ở mục V.

III. Tiền xử lý ảnh

Tiền xử lý là bước quan trọng để có được kết quả nhận dạng chữ chính xác bằng công cụ Tesseract OCR, bởi vì một ảnh poster được chụp bằng tay rất khác so với đầu vào lý tưởng cho OCR:

1) Bức ảnh sẽ có những méo dạng hoàn cảnh (bị nghiêng góc so với phương ngang), trong khi OCR cho rằng bức ảnh chứa chữ được chụp vuông góc với mặt poster.

2) Độ sáng không đồng nhất trên toàn bộ bức ảnh.

3) Trên tờ rơi, thường có nhiều khối chữ, và những khối đó thường không đồng nhất (kiểu chữ, kích cỡ chữ, màu sắc, nền). Công việc tiền xử lý của chúng tôi được thiết kế để giải quyết những vấn đề trên. Điều đó được thể hiện ở những mục nhỏ sau đây.

A. Chỉnh sửa méo dạng hoàn cảnh

Ta giả sử rằng bề mặt của poster/tờ rơi tạo ra một mặt phẳng trong không gian ba chiều, và được biểu diễn trên một tờ giấy hình chữ nhật. Do đó, miễn là ta tìm ra biên các cạnh của poster/tờ rơi trong bức ảnh, thì ta có thể đưa ra một phép biến đổi biến miền tứ giác không đều thành miền hình chữ nhật nằm ngang. Để tìm ra các cạnh của hình tứ giác này, ta thực hiện các bước sau:

1. Dùng phương pháp xác định đường biên Canny để tạo ra một bản đồ cạnh của bức ảnh gốc. Thông số của bộ Canny có thể được cải thiện để thích hợp hơn với mục đích của chúng ta. Vì các đường biên mong muốn phải tương đối đậm và dài, do đó, phương pháp "Ngưỡng gradient" được thiết lập để loại bỏ các đường biên yếu, và bộ lọc phẳng cần được áp dụng để tránh các đường biên được tạo ra từ các kết cấu cục bộ của bức ảnh.

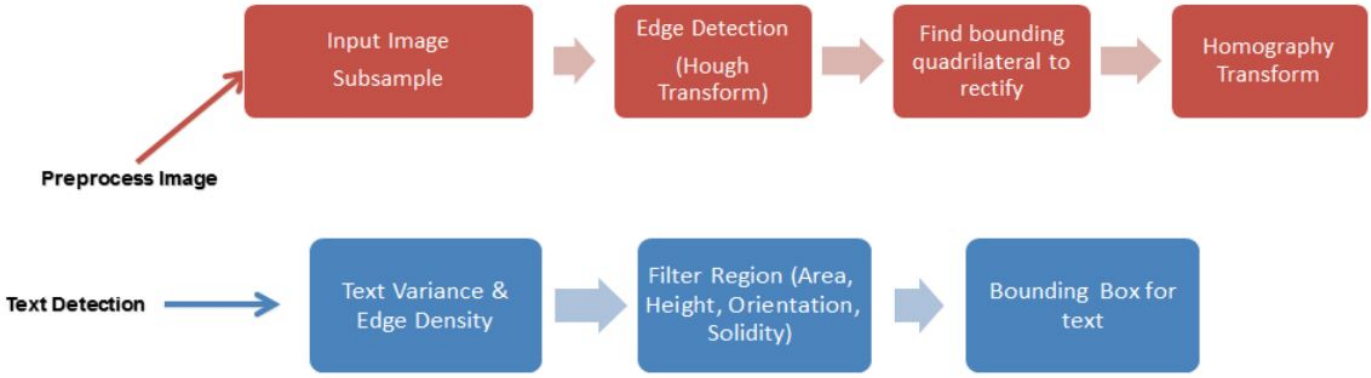
2. Thực hiện biến đổi Hough trên bản đồ đường biên để lấy ra các đường bao có thể.

3. Định vị bốn mặt của các cạnh tứ giác bằng cách phát hiện hai đường biên ngang gần nhau và hai đường biên đứng gần nhau. Việc này được hoàn thành nhờ vào sử dụng các hàm [houghpeaks] và [houghlines]. Vì có thể có nhiều đường biên được xác định, nên chúng ta chọn và phân bậc chúng dựa trên một vài tiêu chuẩn: a. Ta chỉ chấp nhận các đường biên lệch trong khoảng +-15 độ so với các trục chuẩn (đứng và ngang). b. Ta chỉ chấp nhận các đường biên dài hơn 1/4 kích cỡ bức hình. Những đường dài hơn sẽ được ưu tiên hơn. c. Các đường biên nằm càng xa tâm của bức hình sẽ có độ ưu tiên cao hơn các đường biên nằm gần tâm. Bên cạnh đó, một vài phép đảm bảo được đưa vào để giải quyết trường hợp những phần cạnh không được tìm thấy (vì các cạnh nằm bên ngoài vùng được chụp ảnh). Hình 4 biểu thị sự nhận dạng các cạnh có khả năng là đường bao của hình tứ giác.

4. Một khi chúng ta xác định được bốn cạnh bao quanh hình tứ giác, ta suy ta phép biến đổi để biến nội dung bên trong hình tứ giác méo dạng thành một vùng hình chữ nhật chuẩn, điều này giúp ta đạt được sự chuẩn xác hình học. Hình 5 chỉ ra các cạnh bao quanh tứ giác, và hình 6 cho thấy kết quả sau khi xóa bỏ sự méo dạng hoàn cảnh.

B. Phát hiện văn bản và chiết xuất từng khối chữ riêng biệt

Chúng tôi sử dụng bức ảnh đã được chỉnh sửa hình học để phát hiện vùng văn bản. Ta có hai bước chính trong phần này: đầu tiên là tìm vị trí của văn bản; thứ hai là gộp các khu vực văn bản thành các khối chữ riêng biệt. Bằng cách làm đồng nhất từng khối chữ, ta sẽ đạt được kết quả nhận dạng tốt hơn nhiều từ Tesseract. Để tìm ra khu vực có văn bản, chúng tôi sử dụng hai tiêu chuẩn thử nghiệm:



Hình 3. Quy trình xử lý ảnh.

1. Sự thay đổi ảnh cục bộ. Đối với mọi vị trí trong bức ảnh, ta tính ra sự thay đổi độ sáng bên trong một mặt nạ 31 pixel. Những chỗ mà sự thay đổi đó nhỏ hơn một ngưỡng nhất định (10% của giá trị thay đổi lớn nhất) sẽ được xác định là không có văn bản (ví dụ là nền phẳng của bức ảnh).

2. Mật độ cạnh cục bộ. Ta đầu tiên tính một bản độ cạnh dựa trên phương pháp Canny. Sau đó, đối với mọi vụ trí trên bức ảnh, ta tính ra tỉ lệ giữa các pixel cạnh trên tất cả các pixel trong một ma trận gồm 31 pixel, chúng tôi gọi đó là "mật độ cạnh". Những khu vực có mật độ cạnh nằm trong khoảng xác định từ 0.05 đến 0.5 sẽ được coi là văn bản. Các xác định này sẽ phân biệt khu vực văn bản với khu vực có nội dung hình ảnh, nơi có nhiều thông tin cục bộ dẫn đến mật độ cạnh cao hơn những khu vực chứa nội dung văn bản. Hình 7 cho thấy hai phương pháp thử nghiệm được áp dụng cho bức ảnh mẫu.

Sau khi những khu vực có khả năng chứa văn bản được xác định dưới dạng bản độ nhị phân, ta áp dụng toán tử dãn để loại bỏ các lỗ nhỏ, và gom các khu vực có chữ lại dựa trên tính liên kết riêng biệt. Các khu vực được gom lại này, sau đó, tiếp tục được lọc bởi một vài phép thử:

1. Các khu vực có độ định hướng thay đổi trong khoảng 10 độ so với phương ngang thì bị bỏ qua.

2. Các khu vực có diện tích quá to hoặc quá nhỏ đều bị từ chối.

3. Các khu vực có tỉ lệ diện mạo (chiều rộng/chiều dài) nhỏ hơn 2.0 bị bỏ qua (Giả sử chữ nằm ngang).

4. Các khu vực độ vững chắc nhỏ hơn 60% bị từ chối (Những khu vực này thì khó có thể là hình chữ nhật). Hình 8 cho thấy những vùng văn bản được nhận dạng và kết quả cuối cùng sau khi được lọc (Nhãn màu vàng).

Cuối cùng, ta có được những khối văn bản mong muốn. Kết quả được thể hiện trong hình 9.

Ta cũng có thể áo dụng nhiều bước công phu hơn để cải tiến kết quả đầu ra của nhận dạng văn bản, tuy nhiên trong tầm của dự án này, chún tôi nhận thấy kết quả là đủ tốt cho mục đích xác định chính xác thông tin quan trọng của sự kiện từ bức ảnh chụp bằng điện thoại thông minh.

C. Nhị phân hóa

Ta nhị phân hóa mỗi khối văn bản riêng biệt sử dụng phương pháp Otsu, điều này về cơ bản là có thể nhị phân hóa thích nghi với thông tin cục bộ. Chúng tôi ghi nhận rằng Tesseract có khả năng phát hiện cả chữ đen trên nền trắng và ngược lại, do đó, chương trình của chúng tôi không cần phải phân biệt điều này.

IV. Nhận dạng chữ cái quang học

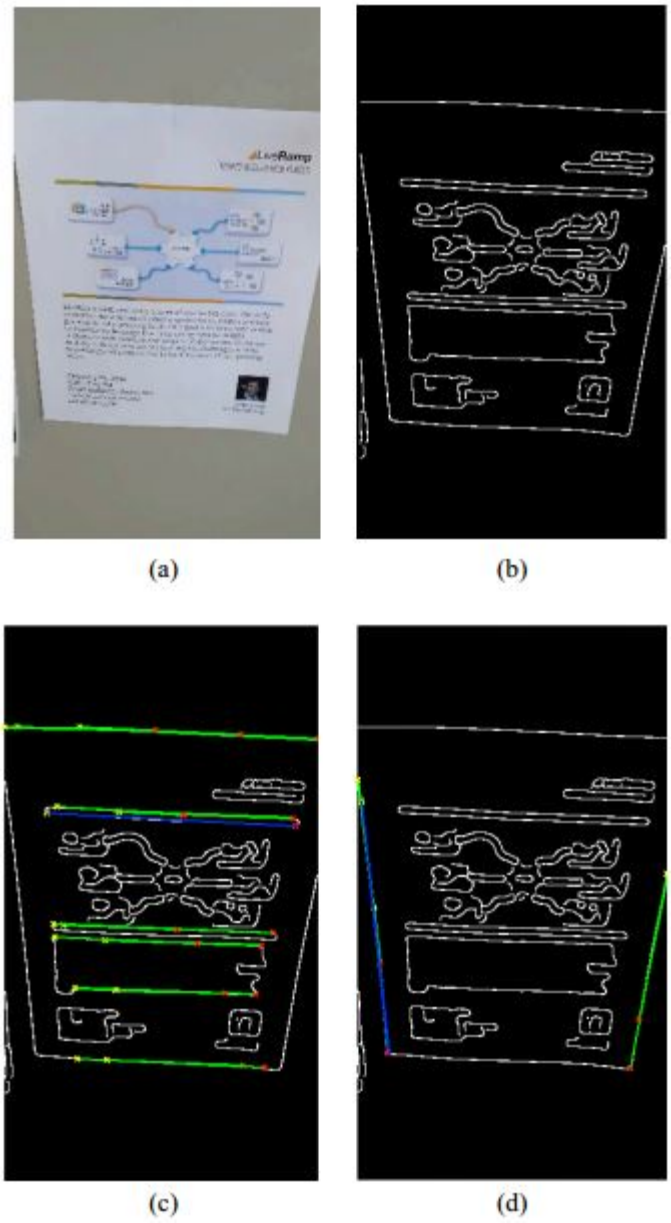
A. Tesseract

Tesseract là nộp công cụ OCR mã nguồn mở, phát triển tại phòng thí nghiệm HP, và hiện tại được duy trì bởi Google. Đây là một trong những công cụ OCR mã nguồn mở chính xác với khả năng đọc một lượng lớn các định dạng hình ảnh và chuyển đổi chúng thành chữ cái với hơn 60 ngôn ngữ [3]. Thư viên Tesseract sẽ được sử dụng trong máy chủ của chúng tôi.

Thuật toán Tesseract sẽ giả thiết đầu vào là một bức ảnh nhị phân và thực hiện bước tiền xử lý của nó trước, theo sau bởi một tầng nhận dạng. Một bộ phân loại thích nghi, tiếp đó, sẽ được dùng để nhận dạng chữ.

B. Máy chủ OCR

Trên máy chủ của chúng tôi, đầu vào của Tesseract là một vài vị trí trong bức ảnh đã được nhận dạng chữ, thay vì toàn bộ bức ảnh. Văn bản kết quả của mỗi vùng, sau đó, được nối với nhau như là ngõ ra cuối cùng của OCR. Mặc dù có thể có một số vùng dương tính giả, nhưng chúng tôi vẫn xem xét đầu ra của OCR là chính xác miễn thông tin về ngày và địa điểm được bao gồm trong văn bản đầu ra cuối cùng.

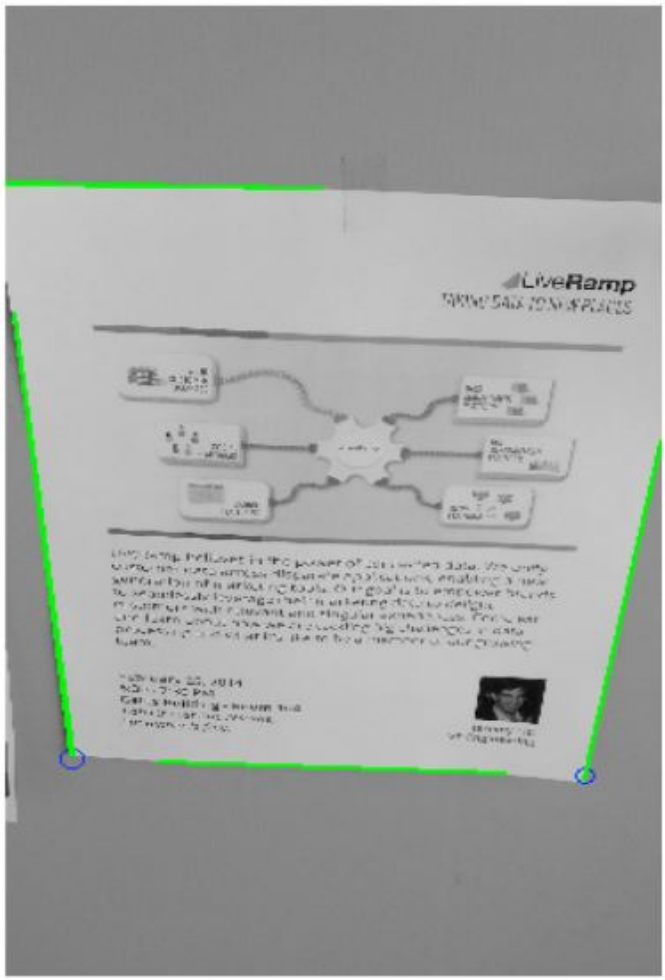


Hình 4. Sử dụng phương pháp Canny để xác định các cạnh có khả năng là đường bao tứ giác. (a) Đầu vào; (b) Kết quả phương pháp Canny; (c) Các cạnh nằm ngang từ biến đổi Hough; (d) Các cạnh thẳng đứng từ biến đổi Hough

V. Hậu xử lý

A. Trích xuất thông tin văn bản

Một khi ta lấy được văn bản từ poster nhờ vào Teseract OCR, chúng ta sẽ muốn trích ra thông tin về thời gian và địa điểm từ văn bản đó. Đối với việc trích xuất thời gian, chúng tôi sử dụng một bộ thư viện Java phân tích cú pháp thời gian [4]. Bộ phân tích cú pháp này sẽ tách tất cả thời gian ra từ chuỗi cho sẵn, sau đó chúng tôi sẽ lọc chúng dựa trên so sánh văn bản và vị trí văn bản, rồi chọn ra thời gian bắt đầu và thời gian kết thúc. Một ví dụ chuẩn về thời gian mà chúng tôi đã chiết xuất được sẽ có dạng "Tue Feb 04 18:30:00 PST 2014".

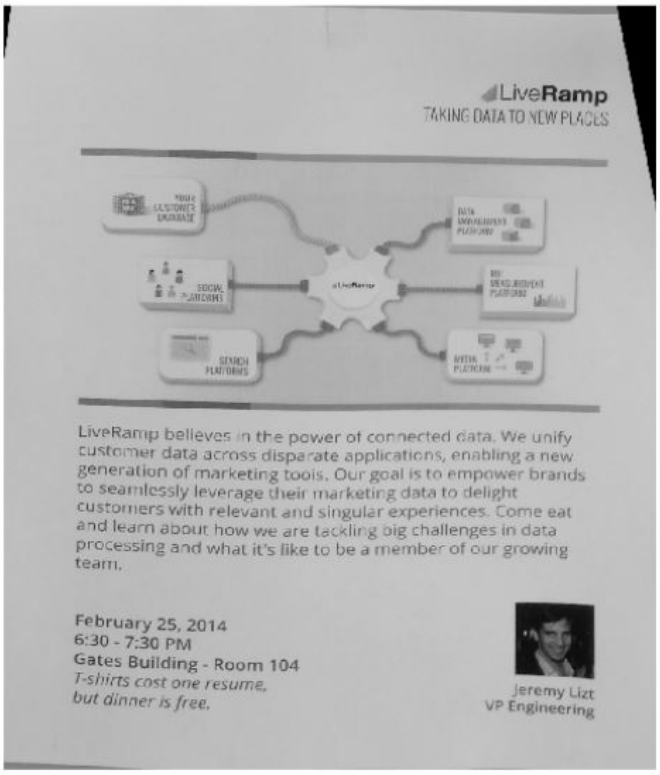


Hình 5. Kết quả lựa chọn đường bao: Viền của poster được vẽ bằng các đường màu xanh lục. Các chấm xanh lam biểu thị cho các góc của poster.

Đối với trích xuất địa điểm, đầu tiên, chúng tôi thực hiện so sánh chuỗi của các địa điểm phổ biến tại khuôn viên Stanford như "Gates", "Packard", "Huang",... Sau đó, chúng tôi sẽ so sánh các từ khóa xác định địa điểm như "tòa nhà", "phòng",... Nếu có kí tự số theo sau các từ khóa trên, chúng tôi sẽ thêm các kí tự số đó vào từ khóa. Nếu không, chúng tôi sẽ chiết ra các từ đứng trước từ khóa và chèn nó vào phần trước của từ khóa. Một ví dụ chuẩn về địa điểm mà chúng tôi đã chiết xuất được sẽ có dạng "Gates Room 104", "Huang Mackenzie Room",...

B. Thêm thông tin của sự kiện vào Google Lịch

Sau khi các thông tin về sự kiện được trích ra từ chuỗi, chúng tôi sẽ thêm nó vào Google Lịch. Đối với người dùng, một khi họ đã chụp ảnh, một cửa sổ lịch sẽ xuất hiện trên điện thoại với thông tin về địa điểm và thời gian được tự động điền vào các ô thông tin của Google Lịch. Hình 11(e) cho thấy giao diện của ứng dụng.



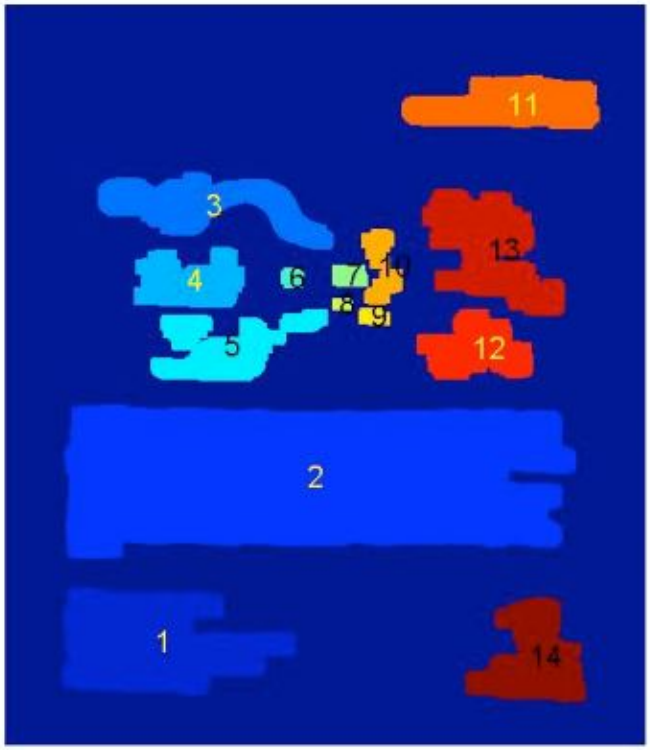
Hình 6. Bức ảnh đã được chỉnh sửa hình học (chỉnh lưu) sau khi tiến hành ánh xạ nội suy.

Hình 7. Bản đồ sự thay đổi độ sáng cục bộ (trái) và mật độ cạnh (phải).

VI. Kết quả

Trong mục này, chúng tôi trình bày ngõ ra của mỗi bước cho một vài mẫu ảnh đầu vào. Hình 11 thể hiện một trường hợp: bức ảnh được chụp trong tư thế méo dạng hoàn cảnh và poster này không đồng dạng.

Với các trường hợp kiểm tra này, chúng ta có thể thấy rằng ở bước thứ nhất thuật toán có khả năng tìm ra tứ giác bao và chỉnh lưu bức ảnh. Ở bước tiếp theo, chúng tôi có thể tìm thấy khu vực của văn bản một cách chính xác. OCR sẽ trả về một kết quả khá tốt nếu chúng tôi có thể chỉ ra khu vực thích hợp chứa văn bản. Cuối



Hình 8. Những khu vực có tiềm năng chứa nội dung văn bản được xác định bằng cách phân tích hai phép thử. Những khu vực với nhãn Vàng là kết quả cuối cùng đã được lọc.

cùng, ứng dụng của chúng tôi cũng có thể tìm ra thời gian và địa điểm chính xác từ văn bản và thêm nó vào Google Lịch.

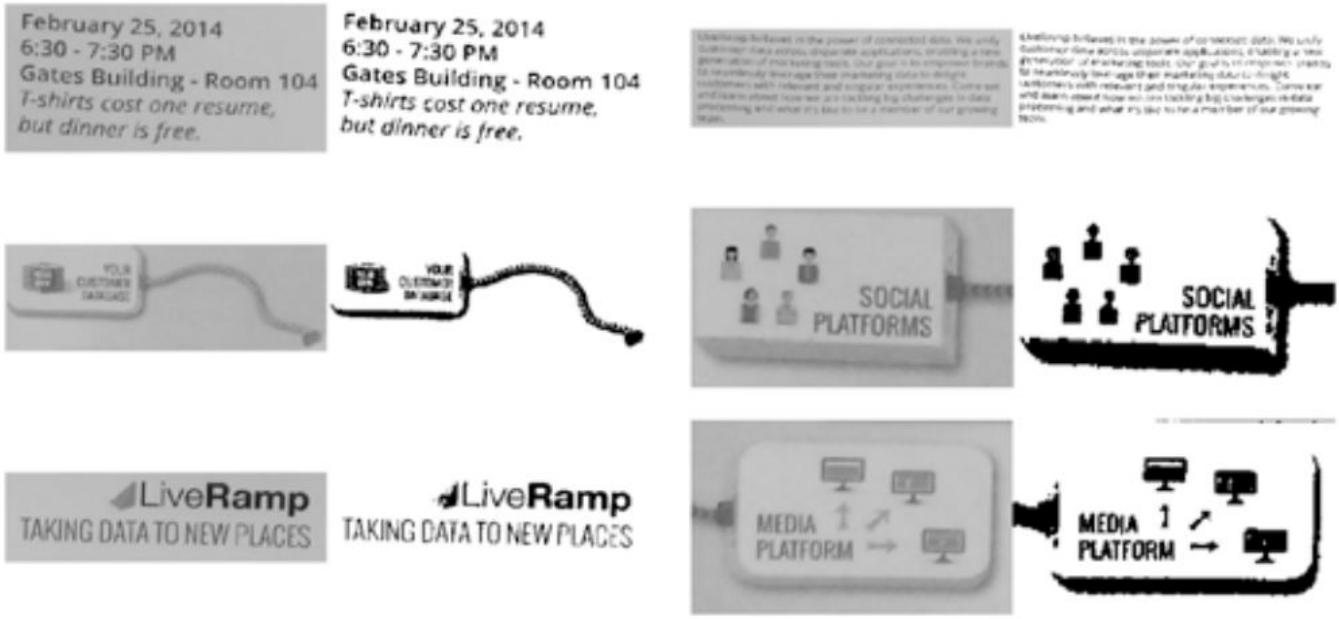
Thuật toán của chúng thôi đôi lúc thất bại khi nền của poster quá phức tạp và chúng tôi chưa thể nhận dạng được khu vực văn bản từ poster, như bức ảnh đầu vào ở hình 10. Chúng tôi cũng chỉnh lưu được bức ảnh. Nhưng vì nền nền của bức ảnh quá lộn xộn, nên chúng tôi chưa tìm được vị trí của phần chứa văn bản dựa trên sự biến đổi độ sáng, mật độ cạnh, hoặc đặc điểm hình học. Do đó, chúng tôi thu được kết quả không chính xác trong trường hợp này.

VII. Công việc tương lai

Trong hệ thống của chúng tôi, ta chỉ lọc ra thời gian và địa điểm từ poster. Nó sẽ thật tuyệt nếu chúng ta có thể tiếp tục đưa ra được chủ đề chính của poster. Chúng tôi có thể xác định chủ để bằng cách xem xét kiểu chữ văn bản, quan tâm đến phần văn bản có kích cỡ chữ lớn hơn. Thêm vào đó, chúng tôi có thể cải tiến sự mạnh mẽ của hệ thống bằng cách áp dụng các kĩ thuật như Biến đổi chiều rộng nét chữ (Stroke Width Transform) để tăng cường khả năng nhận dạng chữ và thực hiện phân đoạn văn bản tốt hơn từ nền hỗn độn hơn là phương pháp Otsu đơn giản.

VIII. Kết luận

Trong dự án này, chúng tôi đã thi công một ứng dụng Android có khả năng chụp một tấm ảnh poster, gửi ảnh lên



Hình 9. Sáu khối chữ riêng biệt được trích xuất sử dụng thông tin khu vực, thể hiện ở hình 8, và ảnh nhị phân tương ứng của chúng.

Hình 10. Một ví dụ ảnh đầu vào mà kết quả đầu ra không chính xác.

máy chủ, tải xuống văn bản đã được xử lý từ máy chủ, lọc ra thông tin sự kiện và cuối cùng thêm nó vào Google Lịch. Rất nhiều phương pháp xử lý ảnh được sử dụng để tiền xử lý bức ảnh, bao gồm Phát hiện cạnh Canny (Canny edge detection), Cân bằng phổ tần số (Histogram equalization), Chỉnh lưu (Rectifying),... Hai phương pháp thử và nhiều bộ lọc được sử dụng để nhận dạng khu vực có chữ. Tesseract được đưa vào để xác định chữ. Và phương pháp Xử lý tiếng nói tự nhiên (Natural language processing) cũng được dùng để trích xuất thông tin sự kiện.

Mặc dù thuật toán của chúng tôi là một phép thử, nhưng với quan sát của chúng tôi thì nó khá mạnh mẽ với các tình huống khó như poster bị che khuất một phần, nền phức tạp, ngữ cảnh chụp khác nhau, độ sáng không đồng nhất,... So sánh với kết quả OCR thô, hệ thống của chúng tôi hiển nhiên có được sự cải tiến về độ chính xác.

TÀI LIỆU

[1] https://stacks.stanford.edu/file/druid:np318ty6250/Sharma_Fujii_Automatic_ContactImporter.pdf

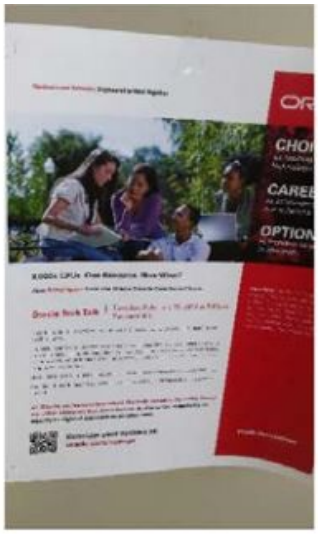
[2] http://www.stanford.edu/class/ee368/Android/Tutorial-3-ServerClient-Communication-for-Android.pdf

[3] https://code.google.com/p/tesseract-ocr/

[4] http://natty.joestelmach.com

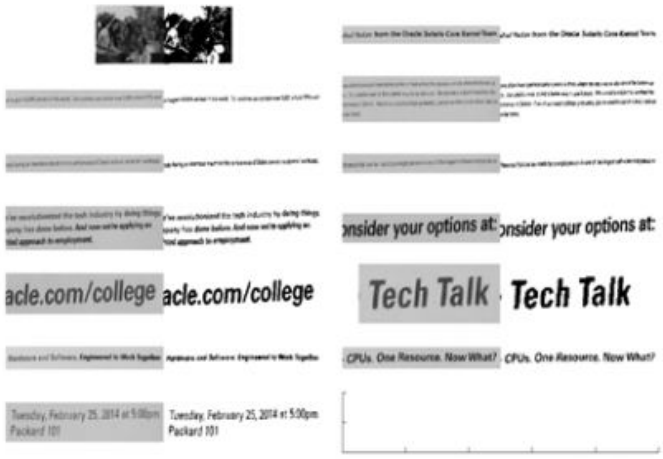
IX. PHỤ LỤC

Công việc được trình bày trong bài báo cáo này cũng mã nguồn là kết quả sự hợp tác của các thành viên sau đây:
* Yang Zhang đề xuất ý tưởng của dự án, và phụ trách các phần: tiền xử lý ảnh, phát hiện và trích xuất những khối ảnh chứa nội dung văn bản. Anh ấy cũng thực hiện video demno với sự giúp đỡ của các thành viên khác trong nhóm.
* Hao Zhang phát triển một bản chế tạo mẫu ứng dụng Android để chạy công cụ Tesseract OCR và Google Lịch. Anh ta cũng xây dựng phần máy chủ cho dự án.

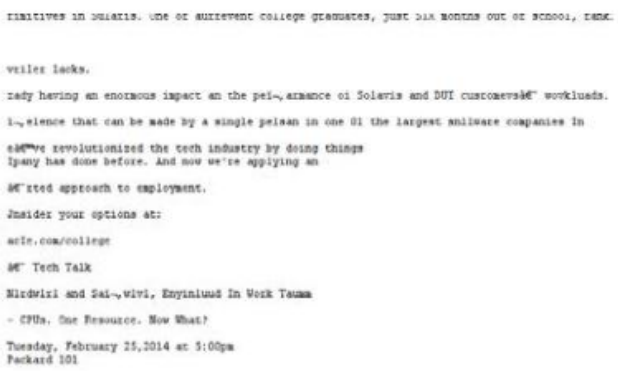


(a) Input Image

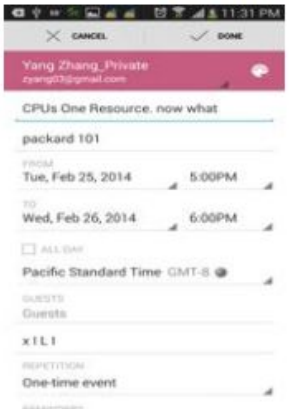
(b) Rectified Image



(c) Text Crop



(d) OCR Result



(e) App Screenshot

Hình 11. Ảnh đầu ra của tất cả các bước. (a) là ảnh đầu vào, (b) là ảnh được chỉnh lưu, (c) là phần cắt chữ, (d) là nội dung văn bản trích xuất được bằng OCR, (e) là giao diện của app.

* Haoran Li thực hiện phần hậu xử lý và làm poster.
* Nỗ lực chung: Hầu hết các phần phân tích và sửa lỗi được phối hợp thực hiện với tất cả các thành viên. Và tất cả các thành viên đều cùng nhau tham gia viết bài báo cáo.