ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN THỊ GIÁC MÁY TÍNH

ĐỀ TÀI: TOP-VIEW CHESS BOARD

Giảng viên hướng dẫn: Ngô Đức Thành

Nhóm sinh viên thực hiện: Lê Trung Hiếu - 18520738

Đặng Hoàng Minh - 18520311

Nguyễn Thanh Điền - 18520600

Phan Khắc Cường - 18520548

HỌC KỲ I NĂM HỌC 2020 - 2021

Lời nói đầu

Vì giới hạn về số lượng trang nên bài báo cáo này sẽ tập trung vào phần các kĩ thuật được áp dụng cho bài toán này, do đó sẽ giả định người đọc đã hoàn thành toán phổ thông cấp 3, kiến thức đại số tuyến tính, lập trình, thị giác máy tính cơ bản, ma trận Homography, hiểu ngôn ngữ lập trình Python và một vài thư viện của ngôn ngữ này như OpenCV, Numpy, Scipy, 3 thư viện này là các thư viện cần thiết, hỗ trợ cho việc hoàn thành đồ án được nhanh hơn.

Phân công giữa các thành viên trong nhóm:

Viết báo cáo: Lê Trung Hiếu
Viết chương trình: Đặng Hoàng Minh và Nguyễn Thanh Điền
Tạo bộ dữ liệu: Phan Khắc Cường

Lời cảm ơn

Nhóm xin chân thành gửi lời cảm ơn đến TS. Ngô Đức Thành – Trưởng khoa Khoa học máy tính, Trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh, đồng thời là giảng viên giảng dạy lớp CS231.L11.KHCL – Nhập môn Thị giác máy tính, đã tận tình giảng dạy cho lớp học.

Mặc dù nhóm đã cố gắng hoàn thành đồ án một cách hoàn thiện nhất, song không thể tránh khỏi những sai sót không mong muốn trong quá trình thực hiện. Nhóm mong rằng sẽ nhận được những đóng góp và nhận xét chân thành từ quý Thầy/Cô, các bạn sinh viên và người đọc để chương trình ngày càng hoàn thiện hơn nữa. Mọi thắc mắc hoặc ý kiến đóng góp xin vui lòng gửi email về một trong các địa chỉ email sau:

18520738@gm.uit.edu.vn (Lê Trung Hiếu)
18520311@gm.uit.edu.vn (Đặng Hoàng Minh)
18520600@gm.uit.edu.vn (Nguyễn Thanh Điền)
18520548@gm.uit.edu.vn (Phan Khắc Cường)

Thành phố Hồ Chí Minh, tháng 1 năm 2021

Nhóm sinh viên thực hiện

Lê Trung Hiếu, Đặng Hoàng Minh, Nguyễn Thanh Điền, Phan Khắc Cường

Mục lục

Lời nói đầu i

Phần 1: Đặt vấn đề 1

Phần 2. Chi tiết phương pháp thực hiện và các kĩ thuật được áp dụng 3

Phần 3. Tập dữ liệu 5

Phần 4. Kết quả thực nghiệm và đánh giá/phân tích chương trình 6

Phần 5. Các công cụ được dùng trong đồ án 11

Tài liệu tham khảo 12

Danh sách các ảnh

Hình 1: Minh họa góc quay giữa hai tuyển thủ trong một trận đấu cờ vua. Hình ảnh được lấy từ video của kênh Youtube ChessBase India, link video: https://www.youtube.com/watch?v=opNjWXsqsKo&t=0s 1

Hình 2: Có một bàn cờ nhỏ đã được thêm vào ở trong video này để thuận tiện cho việc theo dõi bàn cờ. Hình ảnh được lấy từ video của kênh Youtube ChessBase India, link video: https://www.youtube.com/watch?v=6O3WEOIm4m8 2

Hình 3: Minh họa cách làm của nhóm. Một bức ảnh chứa bàn cờ (bên trái) và bàn cờ đã được chuyển hướng nhìn thành top-view (bên phải). 2

Hình 4: Các ảnh minh họa cho 4 phần trong bộ dữ liệu. (a) Ảnh chiếu sáng vừa và hướng camera nghiêng. (b) Ảnh chiếu sáng mạnh và hướng camera từ xa. (c) Ảnh với sánh sáng Mặt Trời và hướng camera người chơi cờ. (d) Ảnh đặc biệt, hướng camera thấp và không bao quát toàn bộ bàn cờ. 5

Hình 5: Các ảnh trong bộ dữ liệu và ảnh top-view của bàn cờ. 9

Hình 6: Bàn cờ thường được hiển thị như thế này trong các trò chơi. ..... **Error! Bookmark not defined.**

Phần 1: Đặt vấn đề

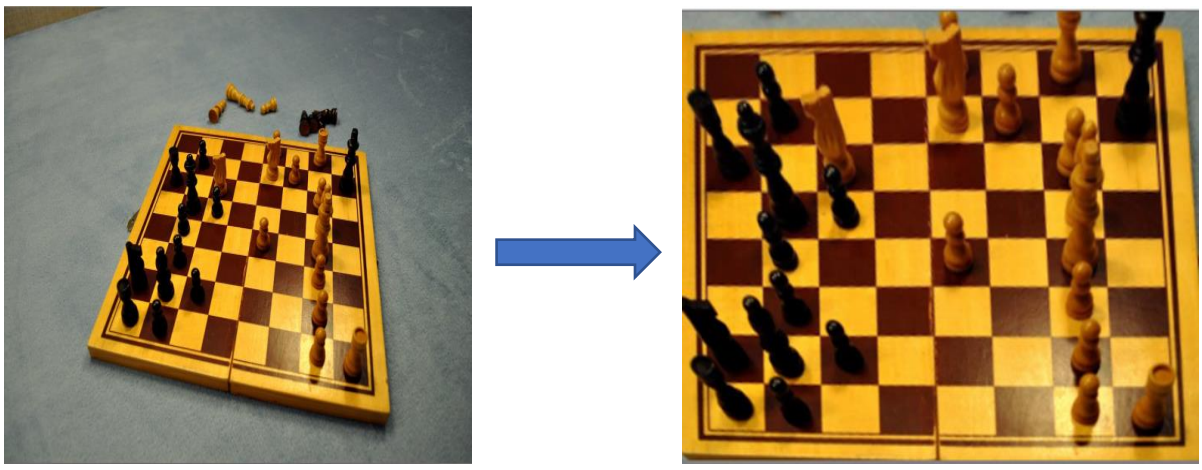
Đối với những người chơi cờ vua hoặc không chơi cờ vua, họ rất có thể sẽ muốn xem các trận đấu cờ. Thông thường, góc quay sẽ ở góc độ sao cho người xem có thể thấy được cả 2 tuyển thủ cùng với tư thế, biểu cảm của họ khi chơi cờ và góc quay thường được giữ nguyên vì một vài lí do nào đó, có thể là giúp người xem không bị rối khi chuyển qua lại giữa các góc quay, hoặc có thể là do kinh phí quá lớn cho việc có hơn 1 máy quay cho mỗi 2 tuyển thủ,…

Hình 1: *Minh họa góc quay giữa hai tuyển thủ trong một trận đấu cờ vua. Hình ảnh được lấy từ video của kênh Youtube ChessBase India, link video: https://www.youtube.com/watch?v=opNjWXsqsKo&t=0s*

Và việc giữ góc quay cố định như vậy đôi khi sẽ khiến người xem khó quan sát được bàn cờ như việc các quân cờ che lấp nhau, khó quan sát được quân cờ đang ở ô nào, độ sáng quá lớn (**Hình 1**),… Cần một giải pháp để có thể giúp người xem có một góc nhìn bàn cờ tốt hơn mà vẫn có thể quan sát được 2 tuyển thủ, như hình sau:

Hình 2: *Có một bàn cờ nhỏ đã được thêm vào ở trong video này để thuận tiện cho việc theo dõi bàn cờ. Hình ảnh được lấy từ video của kênh Youtube ChessBase India, link video: https://www.youtube.com/watch?v=6O3WEOIm4m8*

Trong bài báo cáo này, nhóm sẽ thực hiện công đoạn đầu tiên dùng trong việc biểu diễn bàn cờ trên máy tính như video trên: đó là chuyển bàn cờ trong ảnh (video là một tập hợp các bức ảnh) về thành dạng top-view.



Hình 3: *Minh họa cách làm của nhóm. Một bức ảnh chứa bàn cờ (bên trái) và bàn cờ đã được chuyển hướng nhìn thành top-view (bên phải).*

Phần 2. Chi tiết phương pháp thực hiện và các kĩ thuật được áp dụng

Phương pháp thực hiện của nhóm là như sau:

Tiền xử lý bức ảnh
Chọn ra 4 góc của bàn cờ (ở đây nhóm chọn bằng tay, không phải chọn tự động)
Tính ma trận Homography
Chuyển bàn cờ về dạng top-view

Chi tiết các bước thực hiện là như sau:

Tiền xử lý bức ảnh

Đầu tiên chúng ta luôn cần tiền xử lý bức ảnh vì rất nhiều lý do như ảnh bị mờ, nhiễu, thiếu sáng, quá sáng, có những kênh màu chúng ta không cần dùng đến,… Ở đây, nhóm chỉ thực hiện thay đổi kích thước bức ảnh về 1080:720, nghĩa là chiều rộng 1080 pixel còn chiều cao là 720 pixel. Kích thước 1080:720 được chọn là vì sau nhiều lần thử nghiệm thì có những bức ảnh quá lớn làm cho cửa sổ hiển thị ảnh của cv2.imshow() bị tràn màn hình, dẫn đến khó chọn ra 4 góc bàn cờ. Còn những ảnh có chiều dài hoặc chiều rộng mà nhỏ hơn kích thước tương ứng ở 1080:720 thì sẽ không bị chuyển về kích thước 1080:720 nhằm tránh việc phóng to, sẽ làm mờ ảnh. Vì vậy, kích thước 1080:720 là vừa đủ để ảnh không bị làm mờ mà vẫn đủ lớn để có thể dễ quan sát được bàn cờ trong ảnh. Ngoài ra, ảnh top-view sẽ không bị ảnh hưởng bởi bước tiền xử lý ảnh này vì góc nhìn trong cả 2 ảnh vẫn không đổi, chỉ có kích thước 2 ảnh là bị đổi.

Chọn ra 4 góc của bàn cờ

Chúng ta cần chọn ra 4 góc (điểm) của bàn cờ để có thể xác định được toàn bộ bàn cờ trong bức ảnh. Và 4 điểm này cần phải **được chọn thủ công theo đúng thứ tự trái trên, phải trên, phải dưới và trái dưới** để thuận tiện cho việc xử lý và tính kích thước của bàn cờ (theo pixel). Chúng ta cần chọn ra 4 điểm trong ảnh nhưng nếu 4 điểm này nằm trên quân cờ thì quân cờ sẽ bị cắt mất, chúng ta có thể chọn 4 điểm sao cho không trùng với quân cờ, nhưng nếu vậy thì cần phải chọn sao cho 4 điểm đó vẫn thuộc mặt phẳng mở rộng của bàn cờ, điều này là rất khó để xác định đối với con người, nên hướng camera phải sao cho quân cờ không đè lên 4 cạnh của bàn cờ.

Tính ma trận Homography

Theo như lý thuyết thì càng nhiều corresponding points (tạm dịch: những điểm tương ứng) thì chúng ta càng sẽ có một ma trận Homography cho ra kết quả tốt hơn nếu so với ma trận Homography được tính chỉ với 4 điểm tương ứng (ít nhất phải có 4 điểm để tính ma trận Homography). Chúng ta chỉ có thể chọn ra nhiều điểm tương ứng nếu có 2 bức ảnh cùng được chụp theo một mặt phẳng (camera có thể xoay, vị trí camera có thể thay đổi), nhưng do chúng ta chỉ có mỗi bức ảnh bàn cờ chưa được chuyển về dạng top-

view, nên việc tìm các điểm tương ứng là vô cùng khó khăn. Nhưng rất may là chúng ta luôn xác định được vị trí của 4 góc bàn cờ nằm ở đâu trong cả 2 bức ảnh (ảnh chưa top-view và ảnh đã top-view). Đối với ảnh chưa top-view thì đã được chọn ra từ bước trên. Vì chúng ta chỉ cần ảnh top-view của bàn cờ nên chúng ta có thể cho ảnh top-view của bàn cờ tọa độ lần lượt theo thứ tự sau:

Trái trên: $(0,0)$
Phải trên: $(0, khoảng\ cách\ từ\ điểm\ trái\ trên\ đến\ phải\ trên)$
Phải dưới: $(\ khoảng\ cách\ từ\ điểm\ trái\ trên\ đến\ phải\ trên\ ,$
$khoảng\ cách\ từ\ điểm\ trái\ trên\ đến\ trái\ dưới)$
Trái dưới: $(khoảng\ cách\ từ\ điểm\ trái\ trên\ đến\ trái\ dưới, 0)$
Trong đó, những điểm trái trên, phải trên, phải dưới và trái dưới là những tọa độ của bàn cờ đã được xác định từ bước *Chọn ra 4 góc của bàn cờ.*

Sau khi đã có 4 điểm tương ứng ở 2 bức ảnh thì chúng ta chỉ đơn giản là tính 8 hệ số của ma trận Homography bằng hàm cv2.getPerspectiveTransform() (gọi tắt là getPer từ bây giờ). Hàm này có tên gọi có vẻ không liên quan nhưng thực chất trong OpenCV, hàm này cũng tìm ma trận Homography giống như hàm cv2.findHomography() (gọi tắt là findHomo từ bây giờ), nhưng getPer chỉ cần 4 điểm tương ứng đồng phẳng, điều mà chúng ta đã xác định được ở phía trên, còn findHomo thì cần ít nhất là 4 điểm ở mỗi ảnh mà chúng ta nghĩ là sẽ tốt cho việc tính toán ma trận Homography và sau đó hàm findHomo sẽ tính ra ma trận Homography tốt nhất có thể dựa vào những điểm đã được cung cấp. Ở đây, chúng ta đã xác định được 4 điểm tương ứng đồng phẳng giữa 2 bức ảnh nên không cần dùng đến hàm findHomo.

Chuyển bàn cờ về dạng top-view

Việc cuối cùng chỉ còn là nhân ma trận Homography với từng điểm ảnh của bàn cờ chưa top-view trong hệ tọa độ Homogeneous là chúng ta sẽ có được ảnh top-view của bàn cờ.

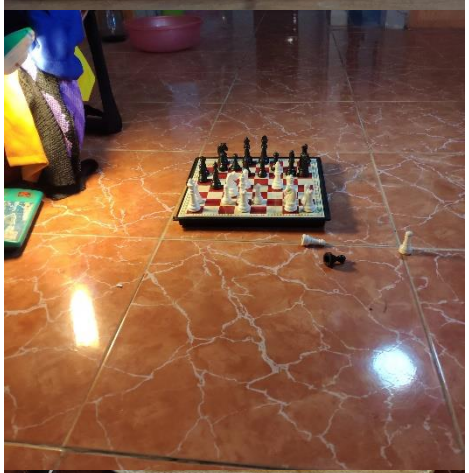
Phần 3. Tập dữ liệu

Bộ dữ liệu được dùng sẽ do nhóm tự tạo để nhằm chọn ra với điều kiện ánh sáng thế nào, hướng camera ra sao sẽ cho ra ảnh top-view bàn cờ dễ quan sát nhất. Tập dữ liệu được nhóm chia ra làm 4 phần, gồm những ảnh được chiếu sáng vừa, chiếu sáng mạnh, điều kiện ánh sáng Mặt Trời và những ảnh đặc biệt (cũng có chia ra những hướng camera khác nhau). Sau đây là một vài ảnh ví dụ:

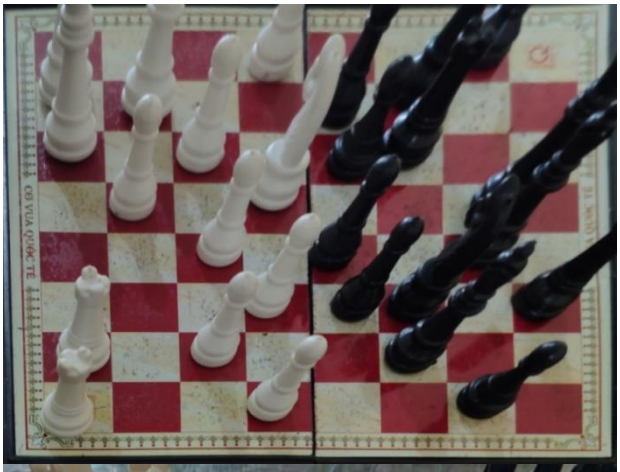
Hình 4: *Các ảnh minh họa cho 4 phần trong bộ dữ liệu. (a) Ảnh chiếu sáng vừa và hướng camera nghiêng. (b) Ảnh chiếu sáng mạnh và hướng camera từ xa. (c) Ảnh với sánh sáng Mặt Trời và hướng camera người chơi cờ. (d) Ảnh đặc biệt, hướng camera thấp và không bao quát toàn bộ bàn cờ.*

Phần 4. Kết quả thực nghiệm và đánh giá/phân tích chương trình

Sau khi đã có chương trình và bộ dữ liệu thì việc cuối cùng sẽ là kiểm tra kết quả dựa trên bộ dữ liệu và đánh giá/phân tích chương trình. Sau đây sẽ là một vài ảnh trong bộ dữ liệu và ảnh top-view tương ứng của bàn cờ sau khi đưa vào chương trình:

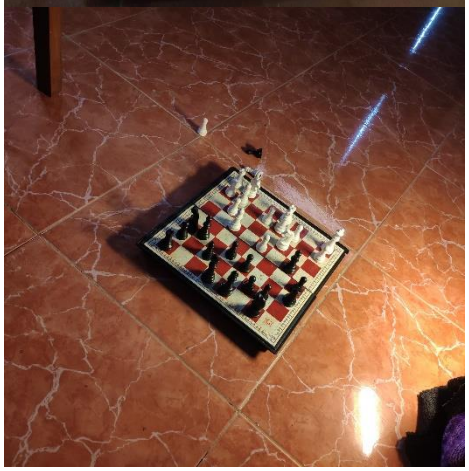
STT	Ảnh trong bộ dữ liệu	Ảnh top-view của bàn cờ
1		
2		
3		

4



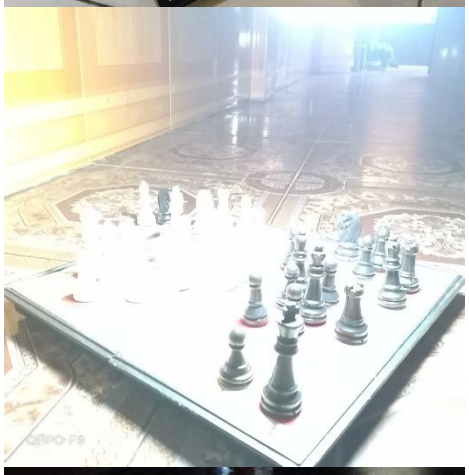
5

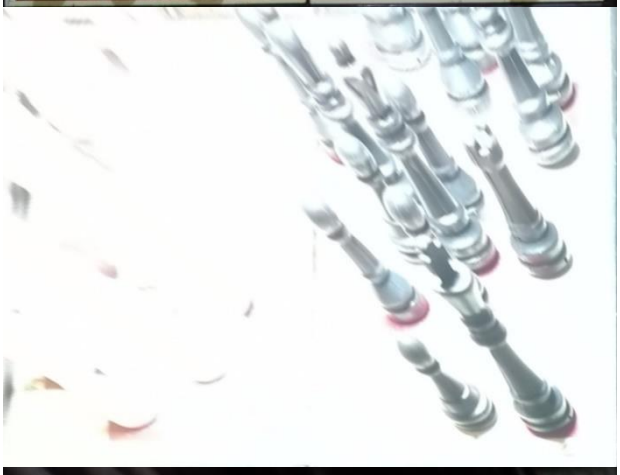
6



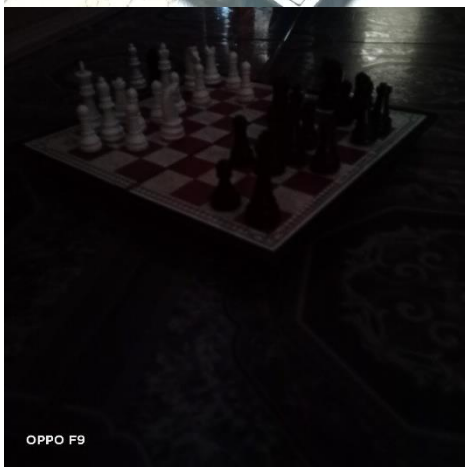
7

8





9



10

11

12

Hình 5: *Các ảnh trong bộ dữ liệu và ảnh top-view của bàn cờ.*

Với những kết quả trên thì có thể dễ dàng nhận thấy rằng ảnh thứ **6** cho ra ảnh top-view dễ quan sát nhất. Bởi vì, xét về hướng camera thì ảnh thứ **6** không có nhiều quân cờ che lấp nhau và cũng chỉ có rất ít quân cờ che mất cạnh bàn cờ, nên khi áp dụng Homography thì chúng ta sẽ nhận được ảnh có top-view tốt hơn so với những hướng camera khác. Còn xét về điều kiện ánh sáng thì ngoại trừ những ảnh quá tối và quá sáng ra thì chúng ta có thể dễ dàng quan sát được bàn cờ nằm ở đâu, từ đó có thể chọn thủ

công ra được 4 góc bàn cờ và có thể quan sát được bàn cờ dưới dạng top-view một cách dễ dàng hơn.

Vậy chỉ cần chúng ta có ảnh với hướng camera “không có nhiều quân cờ che lấp nhau, có rất ít quân cờ che mất cạnh bàn cờ” và điều kiện ánh sáng dễ nhìn thì sẽ có được ảnh bàn cờ dưới dạng top-view dễ quan sát nhất.

Phần 5. Các công cụ được dùng trong đồ án

Ngôn ngữ lập trình: Python, phiên bản 3.8.3

Môi trường lập trình: Jupyter Notebook

Các thư viện và công dụng trong đồ án:

Thư viện OpenCV, phiên bản 4.0.1: được dùng để đọc, hiển thị, thay đổi kích thước của ảnh, lấy tọa độ 4 góc bàn cờ trong ảnh và tính ma trận Homography.

Link website của thư viện OpenCV: https://www.opencv.org/

Thư viện Numpy, phiên bản 1.18.5: được dùng để chứa 4 điểm góc của bàn cờ, nhằm nâng cao tốc độ xử lý của chương trình vì Numpy được xây dựng dựa trên C và Python, việc được xây dựng trên C đã góp phần rất lớn cho tốc độ xử lý của thư viện này.

Link website của thư viện Numpy: https://www.numpy.org/

Thư viện Scipy, phiên bản 1.5.0: được dùng để tính khoảng cách giữa 2 điểm pixel nhằm phục vụ cho việc xác định thứ tự các điểm của 4 góc bàn cờ gồm: trái trên, phải trên, phải dưới và trái dưới.

Link website của thư viện Scipy: https://www.scipy.org/

Tài liệu tham khảo

Ding, J. (2016, January 7). *ChessVision*. Retrieved from GitHub: https://github.com/jialinding/ChessVision/blob/master/board.py

OpenCV. (2018, December 22). *Camera Calibration and 3D Reconstruction*. Retrieved from OpenCV: https://docs.opencv.org/master/da/d54/group__imgproc__transform.html#gae66ba39ba2e47dd0750555c7e986ab85

OpenCV. (2018, December 22). *Geometric Image Transformations*. Retrieved from OpenCV: https://docs.opencv.org/4.0.1/d9/d0c/group__calib3d.html#ga4abc2ece9fab9398f2e560d53c8c9780

Rajnish. (2020, September 8). *Python OpenCV | cv2.circle() method*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/python-opencv-cv2-circle-method/

Rosebrock, A. (2014, August 15). *4 Point OpenCV getPerspective Transform Example*. Retrieved from PyImageSearch: https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/