**<a>Máy tính nén ảnh như thế nào</a>**

**<p><b>Thành phần của một bức ảnh</b></p>**

**<p><b>Độ phân giải</b></p>**

<p>Chúng ta đã nghe nhiều về độ phân giải ảnh, vậy nó có ý nghĩa gì? lấy ví dụ một bức ảnh full HD có độ phân giải 1920\*1080 px, con số 1920\*1080 mang ý nghĩa chiều cao của bức ảnh sẽ có 1920 pixel (px) và chiều rộng có 1080 pixel (px), như vậy tổng số điểm ảnh trên bức ảnh sẽ là 1920 x 1080 = 2.073.600 pixel, độ phân giải càng cao đồng nghĩa với việc ảnh có nhiều điểm ảnh (px), khi điểm ảnh càng nhiều ảnh sẽ trở nên nét hơn, và khi zoom vào chúng sẽ không bị vỡ. Mỗi một pixel trên ảnh sẽ thể hiện một màu sắc, tập hợp chúng lại sẽ tạo thành một bức ảnh hoàn chỉnh. Vậy pixel là gì? và chúng được cấu thành như thế nào? </p>

**<p><b>Cấu tạo pixel</b></p>**

<p>Mỗi pixel sẽ được cấu tạo từ 3 pixel con theo mô hình màu RGB, viết tắt của Red-Green-Blue, mỗi pixel con sẽ nhận giá trị trong khoảng từ 0 đến 255. Mô hình này sẽ có giá trị nhỏ nhất là (0,0,0) khi đó ta thu được màu đen và giá trị lớn nhất là (255,255,255) khi đó ta thu được màu trắng, tương tự (255,0,0) là màu đỏ, (0,255,0) là màu xanh và (0,0,255) là màu xanh dương. Mỗi pixel trên ảnh là sự pha trộn của 3 màu RGB nhận giá trị từ 0 đến 255, như vậy theo tinh toán ta sẽ thu được 255\*255\*255 = 16.581.375 triệu màu. Nếu nhìn lý tưởng mỗi pixel trên ảnh sẽ hiển thị một màu riêng lẻ từ hơn 16 triệu màu này, nhưng thực tế hiện nay chưa có công nghệ nào cho phép tạo ra điều này, đó là lý do tại sao người ta phải pha trộn ba màu RGB theo các cường độ khác nhau để tạo ra các màu khác nhau</p>

**<p><b>Hiển thị pixel</b></p>**

<p>Cùng tìm hiểu về các công nghệ hiển thị ảnh, cụ thể là cách hiển thị các pixel này lên màn ảnh sẽ được thực hiện như thế nào? </p>

<p><b><i>**Công nghệ hiển thị RGB stripe</i></b></p>**

<p>Như đã trình bày ở phần trên, RGB stripe là công nghệ gộp ba pixel con theo mô hình màu RGB thành 1 pixel, ưu điểm của phương pháp này là màu sắc hình ảnh được hiển thị một cách chính xác và trung thực, đây cũng là phương pháp phổ biến được sử dụng trong thực tế</p>

<div class="a">

<img src="pics/**rgb.jpg**" alt="Trulli" width="500" height="400">

<p><i>**Cách sắp xếp điểm ảnh theo ma trận RGB stripe**</p></i>

</div>

<p><b><i>**Công nghệ PenTile RGBG</i></b></p>**

<p>Công nghệ này xuất hiện trên các màn hình Amoled đời đầu của Samsung, thay vì mỗi điểm ảnh gồm 3 điểm ảnh con RGB, ma trận Pentile sẽ đặt điểm ảnh con màu xanh lá ngay sau các điểm ảnh đỏ hoặc xanh dương. Như vậy mỗi pixel theo công nghệ này sẽ chỉ có 2 điểm ảnh con thay vì 3 điểm ảnh con trên ma trận RGB stripe, điều này dẫn đến ảnh hiển thị sẽ có độ phân giải cao hơn so với RGB stripe, nhưng lại bị nhược điểm là ảnh có thể hiển thị sai màu đặc biệt xảy ra ở các góc cạnh màn hình hay viền chữ. Nhưng tại sao các màn hình Amoled của Samsung lại sử dụng ma trận điểm ảnh Pentile, nguyên nhân nằm ở tuổi thọ của màn hình. Thông thường các đi ốt phát màu cho điểm ảnh con trên màn hình Amoled rất khó để kiểm soát cường độ, đặc biệt đối với điểm ảnh con xanh dương, đây là nguyên nhân chính dẫn đến màn hình Amoled đời đầu thường bị ám xanh, khi mà đi ốt phát sáng quá mức, tuổi thọ của đi ốt đó sẽ giảm đi, kéo theo tuổi thọ của màn hình giảm theo. Samsung đã nghĩ ra cách sắp xếp theo ma trận Pentile qua đó làm giảm điểm ảnh con xanh dương, kéo dài tuổi thọ cho màn hình</p>

<p>Vấn đề các điểm ảnh màu hiển thị thiếu chính xác màu ảnh có thể khắc phục bằng cách tăng mật độ điểm ảnh hiển thị trên màn hình, khi mật độ điểm ảnh lên tới khoảng 300 ppi thì những sai sót màu của điểm ảnh rất khó để nhận biết bằng mắt thường</p>

<div class="a">

<img src="pics/**pentile.png**" alt="Trulli" width="500" height="400">

<p><i>**Cách sắp xếp điểm ảnh theo ma trận Pentile**</p></i>

</div>

<p><b>**Mật độ điểm ảnh**</b></p>

<p>Nói thêm một chút về mật độ điểm ảnh trên màn hình, thông số này cho biết số điểm ảnh trên 1 inch kích thước theo chiều dài hoặc chiều rộng của màn hình, chỉ số dpi càng cao có nghĩa là số điểm ảnh trên một đơn vị chiều dài càng cao, điều đó đồng nghĩa với việc các điểm ảnh con sẽ dày đặc lên, và khi đó ảnh sẽ hiển thị rất rõ nét</p>

<div class="a">

<img src="pics/**ppi.jpg**" alt="Trulli" width="500" height="400">

<p><i>**Hình minh họa số điểm ảnh trên 1 inch (ppi)**</p></i>

</div>

**<p><b>Ứng dụng thuật toán k-means trong nén ảnh</b></p>**

**<p><b>Tìm hiểu về cách thức nén ảnh</b></p>**

<p>Nén ảnh hiểu đơn giản là làm giảm dung lượng của bức ảnh, đồng nghĩa với việc giảm số điểm ảnh trên một bức ảnh. Khi nén một bước ảnh đồng nghĩa với việc số điểm ảnh trên mỗi inch (dpi) của bức ảnh sẽ giảm đi. Giả sử một bức ảnh có mật độ điểm ảnh 280 dpi nén xuống còn 70 dpi, lúc này 4 điểm ảnh sẽ giảm xuống còn 1 điểm ảnh, dung lượng và độ chi tiết của chúng cũng sẽ giảm theo. </p>

<p>Về cách thức hoạt động khi nén ảnh, phần mềm nén ảnh sẽ tìm ra những điểm ảnh gần giống nhau và biến chúng thành 1, ví dụ khi ta nén bước ảnh 280 dpi thành 70 dpi, phần mềm nén ảnh sẽ lấy giá trị trung bình của 4 pixel sát nhau có giá trị gần giống nhau, sau đó biến 4 pixel đó thành 1 pixel có giá trị là trung bình của 4 pixel cộng lại</p>

<div class="a">

<img src="pics/**4pixels.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Minh họa giá trị trung bình của 4 điểm màu bằng cách lấy trung bình cộng**</p></i>

</div>

<p>Để hiểu rõ hơn về các bước nén ảnh thử lấy một ví dụ về bức ảnh hoa đồng tiền</p>

<div class="a">

<img src="pics/ **flower.jpg**" alt="Trulli" width="500" height="400">

</div>

<p>Bức ảnh trên có thể chia thành 5 cụm màu cơ bản: cụm màu đen ở vị trí tâm của bông hoa, cụm màu vàng nhạt nằm ở nhụy, cụm màu hồng ở cánh hoa, cụm màu trắng và xám ở nền. </p>

<p>Như đã trình bày ở phần trên, ảnh sẽ được nén bằng cách biến các điểm sát nhau thành một điểm ảnh gộp, các điểm ảnh này sẽ có giá trị gần giống nhau và giá trị của điểm ảnh gộp sẽ là giá trị trung bình của chúng. Mỗi một điểm màu có thể được biểu diễn bằng một vecto trong không gian ba chiều (x1, y1, z1) với x1, y1, z1 nhận giá trị từ 0 đến 255. Trong năm cụm màu của bước ảnh trên, trong mỗi cụm màu có thể chứa tới hàng trăm ngàn màu với các cường độ pha trộn khác nhau, nhưng tựu chung lại thì toàn bộ cụm đó có thể nhận được một giá trị màu duy nhất là giá trị trung bình, và khi đó mắt thường vẫn có thể nhận biết được đó là bức ảnh hoa đồng tiền</p>

<p>Cần lưu ý điểm quan trọng trong việc nén ảnh không phải nằm ở việc tính giá trị trung bình của các cụm màu. Vấn đề nằm ở chỗ làm sao khoanh vùng được từng cụm màu trên bức ảnh, đối với mắt thường thì điều này rất dễ dàng nhưng đối với máy tính thì cần có một cách tiếp cận trình tự theo nhiều bước, và đó là lý do tại sao người ta xây dựng thuật toán phân cụm K-mean clusters với hệ số k ở đây có thể hiểu là số cụm, mean là giá trị trung bình của cụm đó. Ở phần dưới chúng ta sẽ cùng tìm hiểu chi tiết cách thức xây dựng thuật toán phân cụm</p>

**<p><b>Mô phỏng thuật toán phân cụm</b></p>**

**Container**

<p>Giao diện trên mô phỏng cách thuật toán K-means hoạt động, với mỗi lần click chuột lên giao diện chúng ta sẽ vẽ lên nó một ô vuông màu đen, hãy thử click thật nhiều lên giao diện, chúng ta sẽ có các cụm ô vuông màu đen. Tiếp theo là chọn số K tướng ứng với các số cụm và nhấn Random. Chẳng hạn bạn chọn K = 3 khi nhấn Random trên giao điện sẽ xuất hiện 3 điểm màu nằm ở ba vị trí bất kỳ, hay nhấn run liên tiếp một vài lần và các điểm màu này sẽ nhảy vào vị trí trung tâm của từng cụm và đổi màu tất cả các điểm trong cụm theo màu của điểm random</p>

<p>Một lưu ý quan trọng là sau mỗi lần click Run, các điểm màu Random sẽ càng tiến gần đến vị trung tâm của cụm, và khi tiến được vào đúng vị trí trung tâm chúng sẽ dừng lại. Để làm được điều này sau mỗi lần nhấn run, mỗi điểm pixel màu đen sẽ tính khoảng cách của nó đến từng điểm màu random khác nhau, và thực hiện so sánh xem điểm random nào gần nó nhất, các dữ liệu này sẽ được lưu vào một array. Từ dữ liệu nhận được từ array, các điểm random sẽ biết được tất cả các điểm gần chúng nhất, sau đó chúng sẽ thay đổi vị trí của mình vào trung tâm các điểm đó. Quá trình này sẽ tiếp diễn một vài lần, sau đó dừng lại khi đã đi vào được đúng vị trí trung tâm của cụm, vị trí này chính là giá trị trung bình của cụm. Như vậy có thể thấy chìa khoá của chương trình này nằm ở việc tính khoảng cách giữa 2 điểm trong không gian ba chiều (x, y, z), cụ thể là khoảng cách từ các điểm của cụm đến điểm random. Cùng tìm hiểu về công thức tính 2 điểm trong không gian bằng minh hoạ dưới đây</p>

<div class="a">

<img src="pics/**distance.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Công thức tính khoảng cách hai điểm trong không gian**</p></i>

</div>

<p>Dễ dàng thấy khoảng cách giữa hai điểm chính là độ dài cạnh huyền trong tam giác vuông. Giá trị bình phương của chúng sẽ là tổng giá trị 2 cạnh vuông bình phương, độ dài 2 cạnh vuông lần lượt có giá trị là (x2-x1)^2 và (y2-y1)^2. Đối với các điểm màu với ba giá trị RGB (x1, y1, z1) trong không gian thì khoảng cách bình phương của chúng sẽ là tổng bình phương của (x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2. </p>

<p>Về chi tiết phần code chạy chương trình, các bạn có thể tìm trong phần page source của trang. Mình sẽ chỉ giải thích một số function quan trọng của chương trinh</p>

<p>Đầu tiên là hàm event.clientX và event.clientY, hai lệnh này sẽ lấy được tọa độ điểm của con chuột, khi click chuột vào giao diện, tọa độ sẽ được lưu vào array Points, và sau đó phần code HTML sẽ tạo ra một ô vuông màu đen ngay chính vị trí được nhấp chuột</p>

<div class="a">

<img src="pics/**createpoint.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Phần code thêm nội dung điểm vào trang HTML**</p></i>

</div>

<p>Để tạo ra điểm lên phần giao diện của chương trình, trang HTML cần thêm được thuộc tính điểm vào phần nội dung của nó, điều này có thể thực hiện được qua lệnh element.setAttribute()</p>

<div class="a">

<img src="pics/**drawpoint.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Hàm vẽ điểm lên giao diện chương trình**</p></i>

</div>

<p>Trong hàm Run chạy chương trình, có hai phần quan trọng là tính khoảng cách từ một điểm trong cụm đến các điểm random, sau đó tìm ra khoảng cách min nhỏ nhất, cần lưu ý thêm dấu … vào trước 1 array trong Javascript nếu muốn tìm ra giá trị nhỏ nhất trong array</p>

<div class="a">

<img src="pics/**run.PNG**" alt="Trulli" width="500" height="400">

</div>

<p>Dưới đây là Flowchart chạy chương trinh</p>

<div class="a">

<img src="pics/**flowchart.png**" alt="Trulli" width="500" height="400">

<p><i>**Flowchart chạy toàn bộ chương trình**</p></i>

</div>

<p><b>**Sử dụng thư viện Sklearn để nén ảnh**</b></p>

<p>Trong thư viện Sklearn của python có chứa thuật toán K-means, mình sẽ dùng thư viện này để nén bức ảnh hoa đồng tiền ở đầu bài, dưới đây là phần code chi tiết</p>

<div class="a">

<img src="pics/**sklearn.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Phần code sử dụng thư viện sklearn để nén ảnh**</p></i>

</div>

<p>Trong đoạn code trên có một số điểm cần lưu ý. Đầu tiền là hàm imread(), đây là hàm đọc 1 bức ảnh, khi chạy hàm này các giá trị điểm ảnh của bức ảnh sẽ được chuyển thành 1 ma trận 3 chiều, ở lớp đầu tiên hiển thị số điểm ảnh theo chiều cao bức ảnh, lớp thứ 2 hiển thị số điểm ảnh theo từng cột trong một hàng, và cuối cùng chính là giá trị của pixel (x,y,z) chứa trong mỗi hàng và cột đó</p>

<p>Cần lưu ý, để chạy được thuật toán k-means trong thư viện SKlearn, chúng ta cần phải chuyển dữ liệu theo mảng 3 chiều này về mảng 2 chiều bằng hàm reshape(height,width,n). Hàm này chứa các tham số điểm ảnh theo chiều cao (height), chiều rộng (width) và số giá trị trong mỗi điểm ảnh n = 3 (x,y,z) </p>

<div class="a">

<img src="pics/**imread.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Hình mô ta cách thức chương trình đọc giá trị của các điểm ảnh, giả sử có 4 điểm ảnh**</p></i>

</div>

<p>Trong phần code sử dụng thư viện Sklearn, một điểm cần lưu ý là giá trị của n\_clusters, nếu muốn ảnh có dụng lượng nén nhỏ thì phải chọn n nhỏ, ở bức ảnh hoa đồng tiền phần trên theo phân tích có năm cụm màu nên mình sẽ chọn thử n\_clusters=5

Khi đã có các dữ liệu từ thuật toán k-means, việc cần làm tiếp theo là tạo ra một bức ảnh đen với giá trị mỗi điểm ảnh là (0,0,0), bức ảnh này sẽ có số điểm ảnh theo chiều cao và chiều rộng bằng với ảnh ban đầu. Sau cùng, chúng ta sẽ lấy các điểm ảnh theo thông tin thu được từ thuật toán k-means và thay thế vào các điểm ảnh đen trên bức ảnh vừa tạo ra

Cùng chạy đoạn code trên sẽ thu được bức ảnh với dung lượng 26kb, so với ảnh gốc là 189kb thì chúng giảm khoảng 7 lần</p>

<div class="a">

<img src="pics/ **flower2.PNG**" alt="Trulli" width="500" height="400">

<p><i>**Bức ảnh sau khi được nén lại bằng 5 cụm màu**</p></i>

</div>