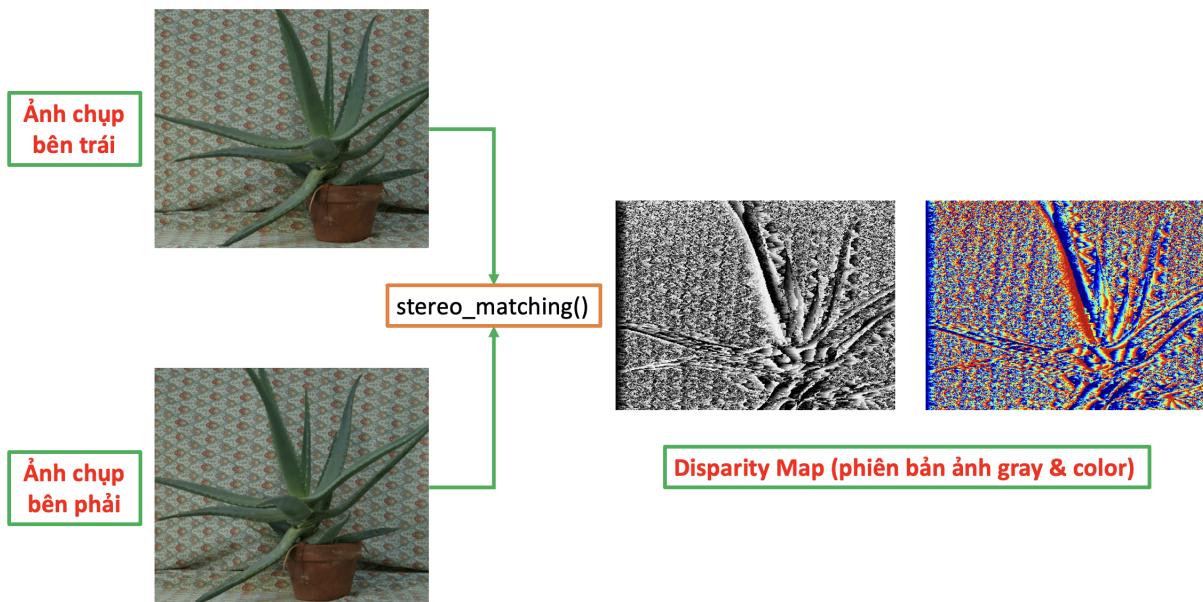


# Depth Information Reconstruction - Project

Ngày 9 tháng 7 năm 2023

## Phần I: Nội dung



Hình 1: Minh họa bài toán Depth Information Reconstruction. Kết quả bài toán là ảnh mô phỏng độ sâu (disparity map) từ một cặp ảnh.

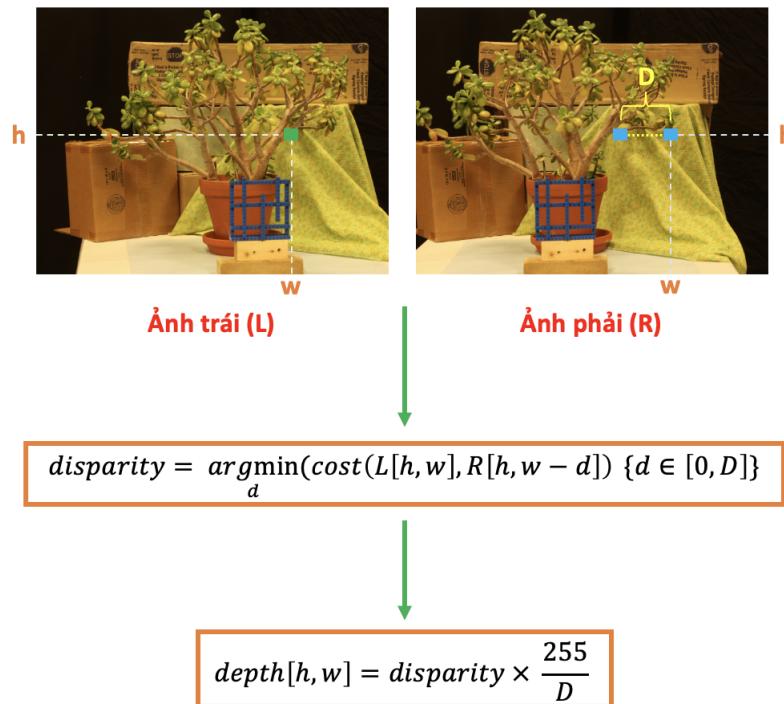
**Note:** Các bạn sẽ thực hiện 5 problem với dữ liệu đầu vào là bộ ảnh **Aloe**. Bộ ảnh này có thể được tải thông qua đường dẫn tại [đây](#).

**Problem 1:** Xây dựng hàm tính disparity map của hai ảnh stereo đầu vào (ảnh bên trái (L) và ảnh bên phải (R)) theo phương thức **pixel-wise matching**. Các bước tính toán trong phương pháp này có thể được miêu tả qua các bước dưới đây:

1. Đọc ảnh chụp bên trái (left) và ảnh chụp bên phải (right) dưới dạng ảnh grayscale (ảnh mức xám) đồng thời ép kiểu ảnh về np.float32.
2. Khởi tạo hai biến height, width có giá trị bằng chiều cao, chiều rộng của ảnh trái.
3. Khởi tạo một ma trận không - zero matrix (depth) với kích thước bằng height, width.

4. Với mỗi pixel tại vị trí  $(h, w)$  (duyệt từ trái qua phải, trên xuống dưới) thực hiện các bước sau:

- Tính cost ( $l1$  hoặc  $l2$ ) giữa các cặp pixel  $left[h, w]$  và  $right[h, w - d]$  (trong đó  $d \in [0, disparity\_range]$ ). Nếu  $(w - d) < 0$  thì gán giá trị  $cost = max\_cost$  ( $max\_cost = 255$  nếu dùng L1 hoặc  $255^2$  nếu dùng L2).
- Với danh sách cost tính được, chọn giá trị  $d$  ( $d_{optimal}$ ) mà ở đó cho giá trị cost là nhỏ nhất.
- Gán  $depth[h, w] = d_{optimal} \times \frac{255}{disparity\_range}$ .



Hình 2: Các bước tính toán giá trị disparity tại vị trí của một pixel bất kì theo phương pháp pixel-wise matching

Dựa vào mô tả trên, khi xây dựng hàm tính stereo matching trong bài tập này, các bạn sẽ cần thực hiện thêm một số yêu cầu sau:

- Thiết kế một hàm có tên gọi **pixel\_wise\_matching()** với tham số đầu vào là:
  - left\_img**: đường dẫn đến ảnh chụp bên trái.
  - right\_img**: đường dẫn đến ảnh chụp bên phải.
  - disparity\_range**: độ dài tối đa của vùng tìm kiếm giá trị disparity tại mỗi pixel. Mặc định sẽ có giá trị là 16.
  - compute\_type**: tên phương pháp tính khoảng cách giữa pixel người dùng muốn sử dụng, giá trị có thể là 'l1' hoặc 'l2'. Mặc định sẽ được gán là 'l1'.
  - save\_result**: giá trị boolean đại diện cho việc có lưu disparity map hay không? Mặc định sẽ có giá trị là True.

Cuối cùng, kết quả trả về của hàm sẽ là một numpy.ndarray đại diện cho disparity map tính được.

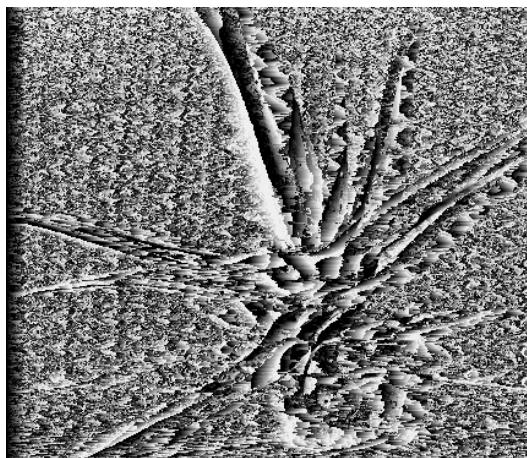
- Khi bắt đầu khởi động hàm, in dòng thông báo "**Compute disparity map using pixel-wise matching with <tên phương pháp người dùng chọn>...**".
- Khi lưu kết quả, ta in ra màn hình dòng chữ "**Saving result...**" và lưu ý rằng sẽ lưu cả hai phiên bản ảnh grayscale và ảnh color map của disparity map.
- Sau khi kết thúc toàn bộ quá trình tính toán, in ra dòng chữ "**Done.**".

Kết quả minh họa khi chạy hàm **pixel\_wise\_matching()** với cặp ảnh Aloe\_1 sẽ có kết quả trả về như sau (các dòng comment tương ứng cho nội dung trả về):

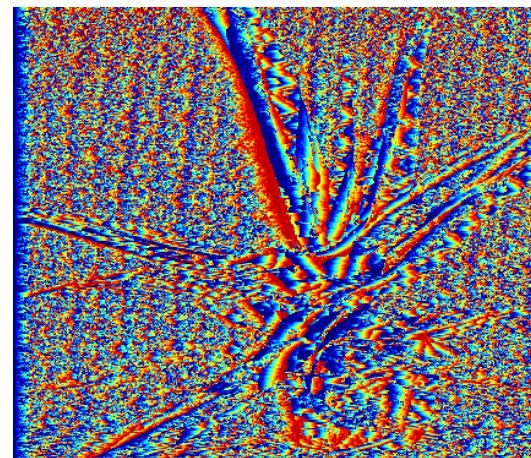
```

1 left_img_path = 'Aloe/Aloe_left_1.png'
2 right_img_path = 'Aloe/Aloe_right_1.png'
3 disparity_range = 16
4
5 pixel_wise_result = pixel_wise_matching(left_img_path, right_img_path, disparity_range
   , compute_type='11', save_result=True)
6
7 # Compute disparity map using pixel-wise matching with L1...
8 # Saving result...
9 # Done.
10
11 pixel_wise_result = pixel_wise_matching(left_img_path, right_img_path, disparity_range
   , compute_type='12', save_result=True)
12
13 # Compute disparity map using pixel-wise matching with L2...
14 # Saving result...
15 # Done.

```

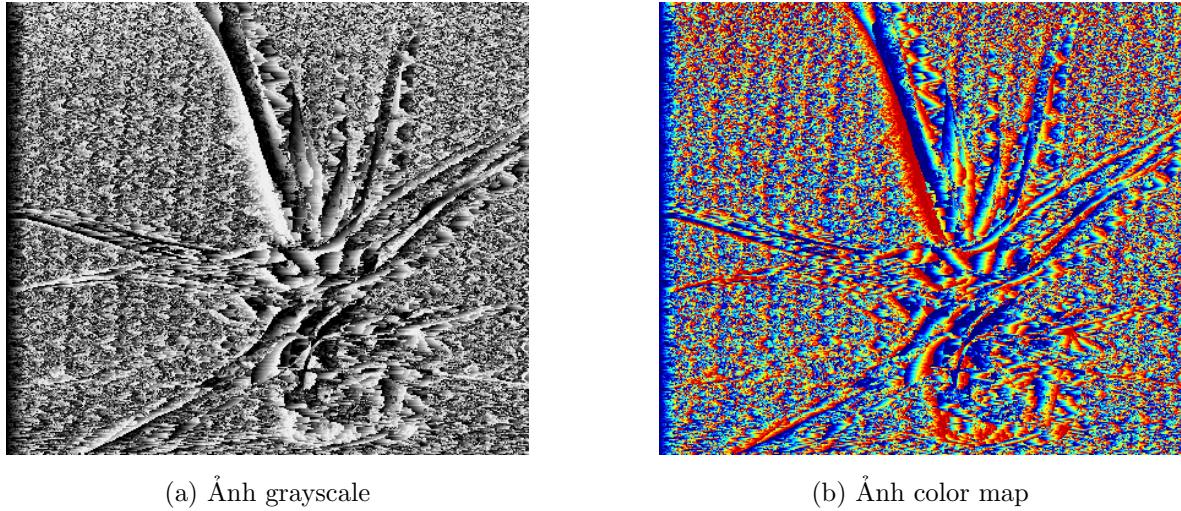


(a) Ảnh grayscale



(b) Ảnh color map

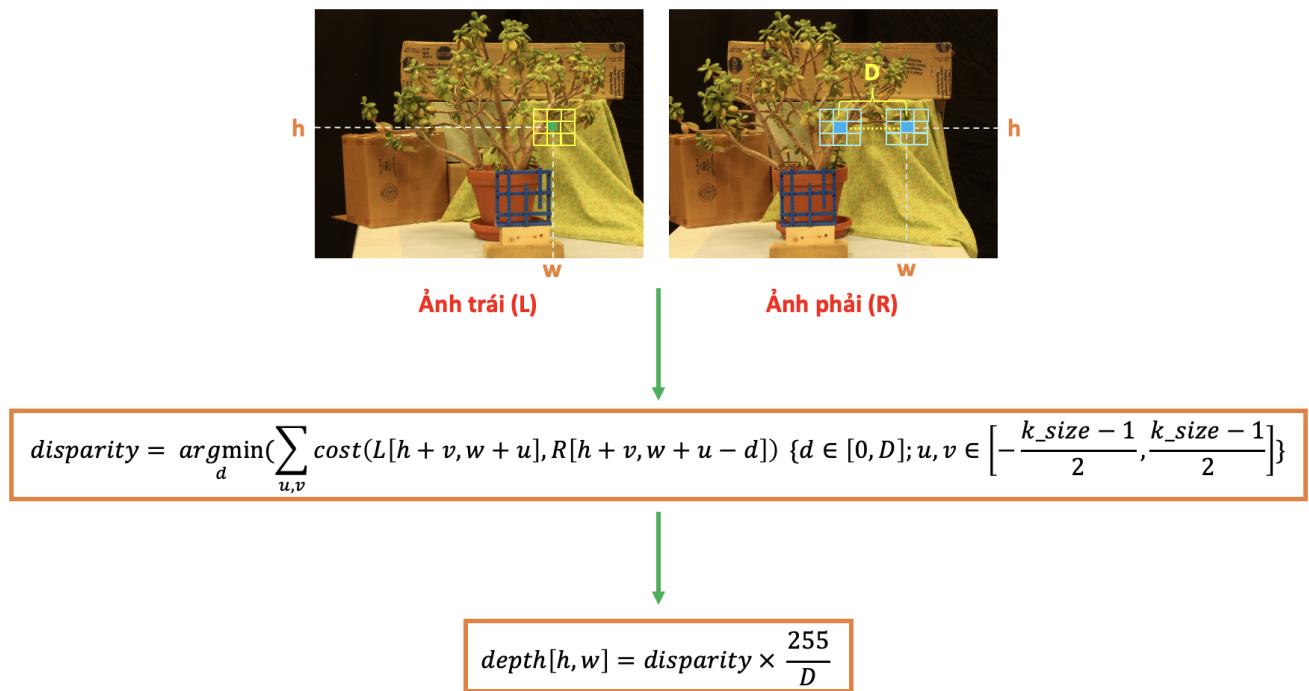
Hình 3: Kết quả disparity map của phương pháp pixel-wise matching sử dụng L1



Hình 4: Kết quả disparity map của phương pháp pixel-wise matching sử dụng L2

**Problem 2:** Xây dựng hàm tính disparity map của hai ảnh stereo đầu vào (ảnh bên trái (L) và ảnh bên phải (R)) theo phương thức **window-based matching**. Các bước tính toán trong phương pháp này có thể được miêu tả qua các bước dưới đây:

1. Đọc ảnh chụp bên trái (left) và ảnh chụp bên phải (right) dưới dạng ảnh grayscale (ảnh mức xám) đồng thời ép kiểu ảnh về np.float32.
2. Khởi tạo hai biến height, width có giá trị bằng chiều cao, chiều rộng của ảnh trái.
3. Khởi tạo một ma trận không - zero matrix (depth) với kích thước bằng height, width.
4. Tính nửa kích thước của window tính từ tâm đến cạnh của window (có kích thước k x k) theo công thức  $\text{kernel\_half} = \frac{k-1}{2}$  (lấy nguyên).
5. Với mỗi pixel tại vị trí (h, w) ( $h \in [\text{kernel\_half}, \text{height} - \text{kernel\_half}]$ ,  $w \in [\text{kernel\_half}, \text{width} - \text{kernel\_half}]$ ; duyệt từ trái qua phải, trên xuống dưới), thực hiện các bước sau:
  - (a) Tính tổng các cost (l1 hoặc l2) giữa các cặp pixel left[ $h + v, w + u$ ] và right [ $h + v, w + u - d$ ] (trong đó  $d \in [0, \text{disparity\_range}]$  và  $u, v \in [-\text{kernel\_half}, \text{kernel\_half}]$ ) nằm trong vùng window với tâm là vị trí của pixel đang xét. Nếu tại vị trí cho  $(w + u - d) < 0$  thì gán giá trị cost của cặp pixel đang xét =  $\text{max\_cost}$  ( $\text{max\_cost} = 255$  nếu dùng L1 hoặc  $255^2$  nếu dùng L2).
  - (b) Với danh sách cost tính được, chọn giá trị d ( $d_{\text{optimal}}$ ) mà ở đó cho giá trị cost tổng là nhỏ nhất.
  - (c) Gán  $\text{depth}[h, w] = d_{\text{optimal}} \times \frac{255}{\text{disparity\_range}}$ .



Hình 5: Các bước tính toán giá trị disparity tại vị trí của một pixel bất kì theo phương pháp window-based matching

Dựa vào mô tả trên, khi xây dựng hàm tính stereo matching trong bài tập này, các bạn sẽ cần thực hiện thêm một số yêu cầu sau:

- Thiết kế một hàm có tên gọi **window\_based\_matching()** với tham số đầu vào là:
  - **left\_img**: đường dẫn đến ảnh chụp bên trái.
  - **right\_img**: đường dẫn đến ảnh chụp bên phải.
  - **disparity\_range**: độ dài tối đa của vùng tìm kiếm giá trị disparity tại mỗi pixel. Mặc định sẽ có giá trị là 16.
  - **compute\_type**: tên phương pháp tính khoảng cách giữa pixel người dùng muốn sử dụng, giá trị có thể là 'l1' hoặc 'l2'. Mặc định sẽ được gán là 'l1'.
  - **kernel\_size**: tham số về kích thước  $k$  của window. Mặc định sẽ có giá trị là 5.
  - **save\_result**: giá trị boolean đại diện cho việc có lưu disparity map hay không? Mặc định sẽ có giá trị là True.

Cuối cùng, kết quả trả về của hàm sẽ là một numpy.ndarray đại diện cho disparity map tính được.

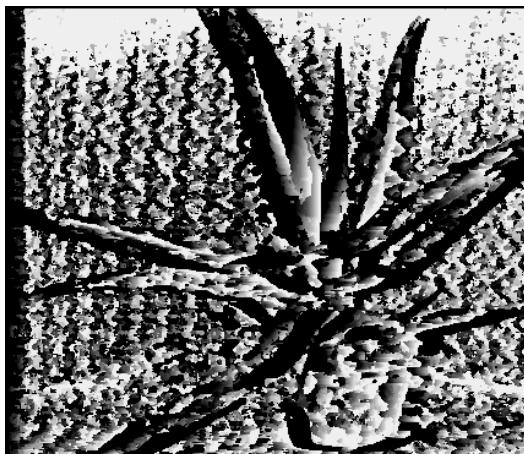
- Khi bắt đầu khởi động hàm, in dòng thông báo "**Compute disparity map using window-based matching with <định danh phương pháp người dùng chọn>...**".
- Khi lưu kết quả, ta in ra màn hình dòng chữ "**Saving result...**" và lưu ý rằng sẽ lưu cả hai phiên bản ảnh grayscale và ảnh color map của disparity map.
- Sau khi kết thúc toàn bộ quá trình tính toán, in ra dòng chữ "**Done.**".

Kết quả minh họa khi chạy hàm **window\_based\_matching()** với cặp ảnh Aloe\_1 sẽ có kết quả trả về như sau (các dòng comment tương ứng cho nội dung trả về):

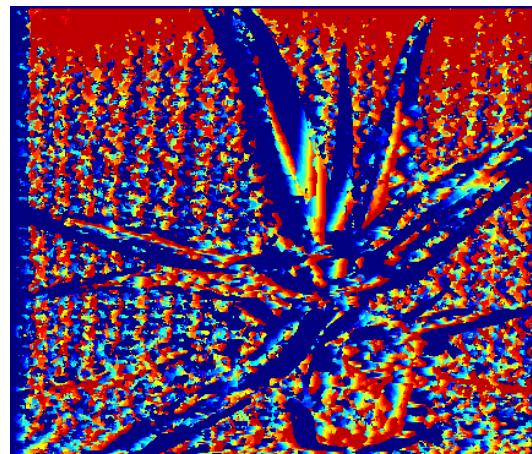
```

1 left_img_path = 'Aloe/Aloe_left_1.png'
2 right_img_path = 'Aloe/Aloe_right_1.png'
3 disparity_range = 16
4 kernel_size = 5
5
6 window_based_result = window_based_matching(left_img_path, right_img_path,
    disparity_range, compute_type='11', kernel_size=kernel_size, save_result=True)
7
8 # Compute disparity map using window-based matching with L1...
9 # Saving result...
10 # Done.
11
12 window_based_result = window_based_matching(left_img_path, right_img_path,
    disparity_range, compute_type='12', kernel_size=kernel_size, save_result=True)
13
14 # Compute disparity map using window-based matching with L2...
15 # Saving result...
16 # Done.

```

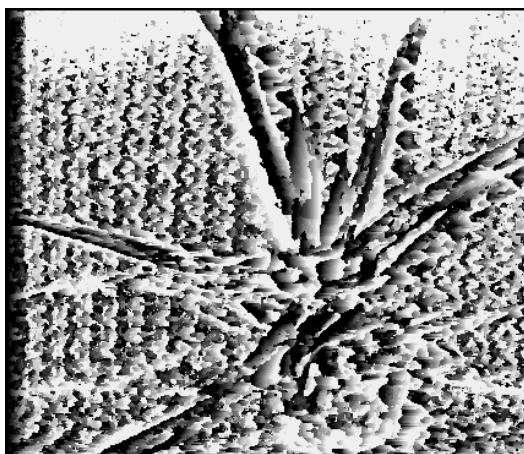


(a) Ảnh grayscale

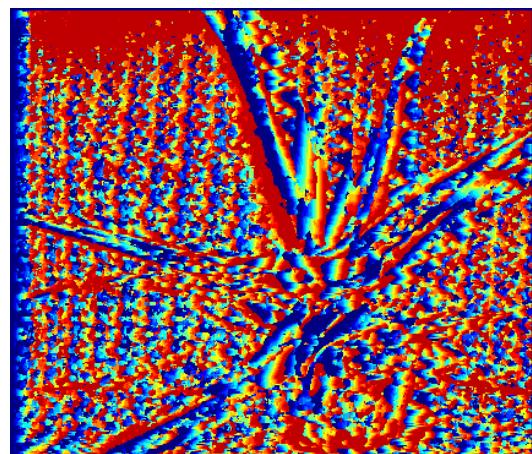


(b) Ảnh color map

Hình 6: Kết quả disparity map của phương pháp window-based matching sử dụng L1



(a) Ảnh grayscale



(b) Ảnh color map

Hình 7: Kết quả disparity map của phương pháp window-based matching sử dụng L2

**Problem 3:** Khi sử dụng hàm tính disparity map đã xây dựng ở Problem 2 cho cặp ảnh Aloe\_left\_1.png và Aloe\_right\_2.png với tham số đầu vào disparity\_range = 64 và kernel\_size = 3 ở cả hai dạng compute\_type, ta được kết quả disparity map như ảnh minh họa sau:



(a) Ảnh chụp bên trái

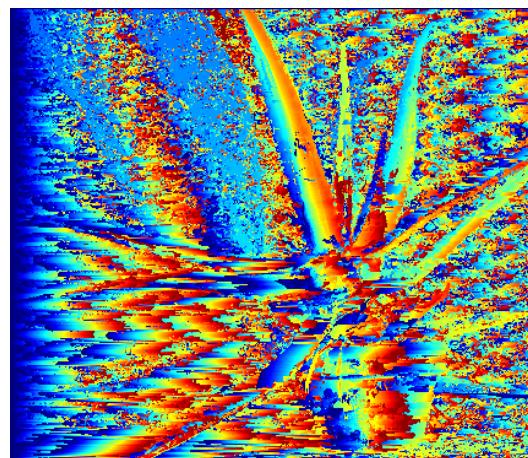


(b) Ảnh chụp bên phải

Hình 8: Ảnh stereo đầu vào của Problem 3



(a) Ảnh grayscale



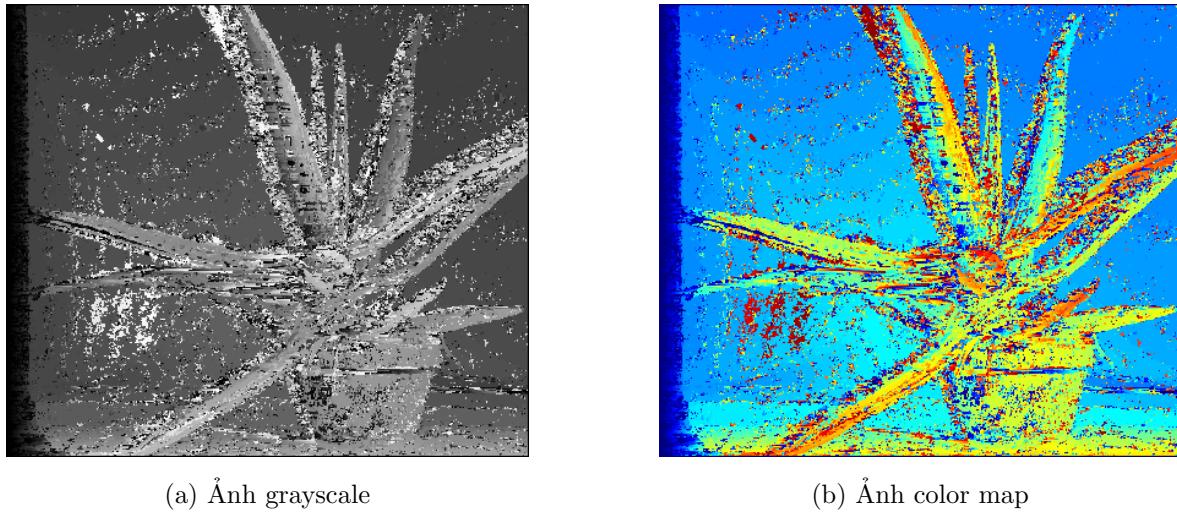
(b) Ảnh color map

Hình 9: Kết quả disparity map của phương pháp window-based matching với cài đặt disparity\_range = 64 và kernel\_size = 3

Có thể thấy với sự thay đổi của các giá trị tham số đầu vào như trên, kết quả disparity map đã phần nào tệ đi (bị nhiễu). Các bạn hãy sử dụng code ở Problem 2 để tạo ra ảnh disparity map với cài đặt này và giải thích (sử dụng markdown) vì sao lại xảy ra kết quả như vậy.

**Problem 4:** Dựa trên hàm tính disparity map theo phương thức window-based matching ở Problem 2 và coi các window là các vector, hãy cài đặt Cosine Similarity trong việc tính sự tương quan giữa hai pixel ảnh trái phải để giải quyết vấn đề ở Problem 3. Công thức Cosine Similarity được mô tả như sau:

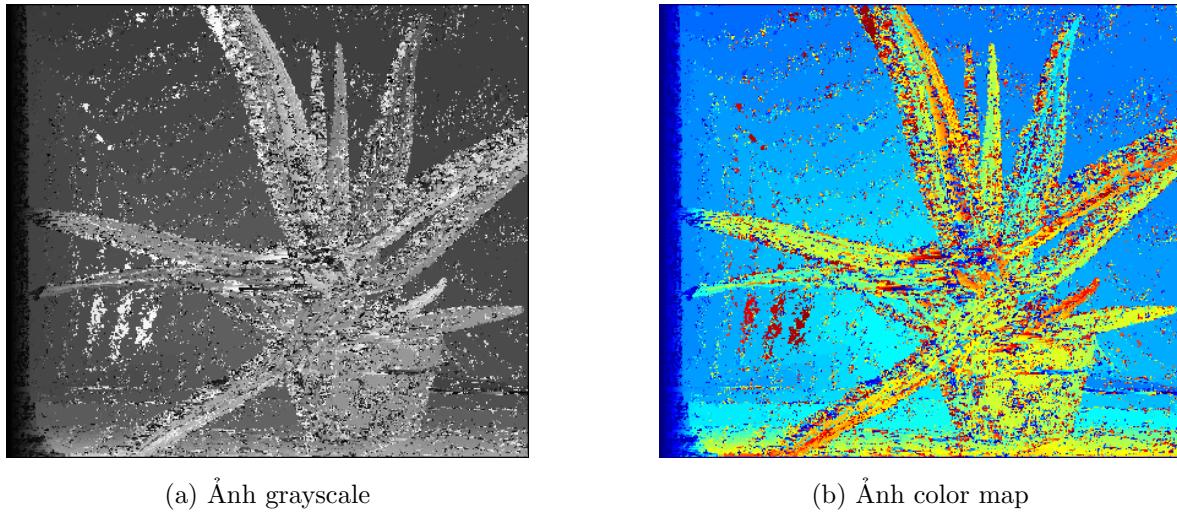
$$\text{cosine\_similarity}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n y_i^2}}$$



Hình 10: Kết quả disparity map của phương pháp window-based matching sử dụng Cosine Similarity với cài đặt tham số và cặp ảnh đầu vào ở Problem 3

**Problem 5:** Dựa trên hàm tính disparity map theo phương thức window-based matching ở Problem 2 và coi các window là các vector, hãy cài đặt Correlation Coefficient trong việc tính sự tương quan giữa hai pixel ảnh trái phải để giải quyết vấn đề ở Problem 3. Công thức Correlation Coefficient được mô tả như sau:

$$\text{correlation\_coefficient}(\vec{x}, \vec{y}) = \frac{n \sum_i x_i y_i - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n \sum_i x_i^2 - (\sum_i x_i)^2} \sqrt{n \sum_i y_i^2 - (\sum_i y_i)^2}}$$



Hình 11: Kết quả disparity map của phương pháp window-based matching sử dụng Correlation Coefficient với cài đặt tham số và cặp ảnh đầu vào ở Problem 3

## Phần III: Trắc nghiệm

1. Ứng dụng nào sau đây có thể liên quan đến việc sử dụng disparity map?
    - (a) Object Detection.
    - (b) Image Classification.
    - (c) Text Retrieval.
    - (d) Depth Estimation for 3D Reconstruction.
  2. Lý do nào sau đây là lợi ích trong việc sử dụng phương pháp window-based matching so với phương pháp pixel-wise matching?
    - (a) Tốc độ tính toán và xử lý nhanh hơn.
    - (b) Hoạt động tốt trên độ phân giải ảnh thấp.
    - (c) Kết quả disparity map mượt hơn.
    - (d) Loại bỏ được bước Calibration trong hệ thống.
  3. Câu nào sau đây mô tả một điểm yếu của phương pháp pixel-wise matching?
    - (a) Yêu cầu ảnh có độ phân giải cao.
    - (b) Độ phức tạp thuật toán cao.
    - (c) Kết quả disparity map bị nhiễu cao.
    - (d) Nhạy cảm với điều kiện ánh sáng.
  4. Vai trò của Disparity Range là gì?
    - (a) Kích thước của search window.
    - (b) Vùng tìm kiếm giá trị disparity.
    - (c) Định nghĩa hàm cost.
    - (d) Ảnh hưởng đến độ sáng của disparity map.
  5. Trong phương pháp window-based matching, kích thước window sẽ ảnh hưởng đến điều gì?
    - (a) Độ chính xác của disparity map.
    - (b) Độ sáng của disparity map.
    - (c) Tốc độ của thuật toán.
    - (d) Độ sâu của disparity map.
  6. Lý do nào sau đây giải thích việc nhân disparity value với tỉ số  $\frac{255}{D}$ ?
    - (a) Cải thiện độ phân giải.
    - (b) Tăng cường độ chính xác.
    - (c) Tăng độ sáng ảnh.
    - (d) Dùng trong việc visualization kết quả.

- H t -