

q1

برای روشن تر شدن عکس و افزایش کیفیت آن تصمیم داریم **intensity** هر پیکسل را ضرب در عددی کنیم تا اختلاف رنگ پیکسل ها مشهود تر شود و همچنین **intensity** شان افزایش یابد. به شرطی که سرریز نشوند. یعنی اگر **intensity** پیکسل در یک چنل برابر با x بود آنگاه آن را به Cx تبدیل کنیم. عدد C مناسب را با آزمون و خطا 5 در نظر میگیریم. ماتریسی که تمام داریه های آن برابر $C=5$ است را میسازیم و آن C مینامیم. ماتریس دیگری با نام **White** میسازیم که تمام درایه های آن مقدار 255 را دارند. اکنون با **for** در هر 3 چنل، C را در ماتریس تصویرمان ضرب کرده و مینیمیم ماتریس حاصل و **White** را محاسبه میکنیم. چون در صورت بیشتر شدن مقدار Cx از 255 مقدار آن پیکسل از چنل **overflow** میشود و رنگبندی عکس بهم میریزد. در انتها عکس را با تابع آماده ی **medianBlur** اندکی تار میکنیم تا نویز های عکس کمتر شوند. میتوانستیم عملیات را به گونه ی دیگری در مختصات **HSV** انجام دهیم ولی نتیجه ی روش پیاده سازی شده در عین سادگی رضایت بخش تر بود.

q2

ابتدا عکس مورد نظر را به **HSV** تبدیل میکنیم و هر یک از چنل ها را تفکیک کرده و در متغیری با نام مختص خودش نگه میداریم تا کار با چنل های آن آسان تر شود. اولین چنل، **hue** یا رنگ بندی را تعیین میکند و طیف آن به صورت زیر است:



هر رنگ در یک بازه ی مشخصی از **hue** قرار دارد که در 0 تا 179 قرار دارد. با آزمون و خطا بر روی عکس داده شده بهترین بازه ی مورد نظر **hue** برای انتخاب گل را در هر عکس یافتیم.

برای عکس اول: بازه ی **hue** برای **Yellow.jpg** را $[20, 30]$ در نظر گرفتیم. از آنجایی که برخی ساقه و برگ ها هم در این طیف قرار گرفتند با تعیین بازه برای **saturation**، سعی میکنیم که خطا را کمتر کنیم. به دنبال تابع خطی ای میگردیم تا به ازای هر مقدار **hue** یک بازه ی مشخصی از **saturation** را تغییر دهیم. این بازه به صورت $[-5hue + 200, -5hue + 350]$ با آزمون و خطا بدست میاید. اکنون **hue** مربوط به تمام پیکسل هایی که شرایط گفته شده را دارند و درون بازه قرار میگیرند را برابر با 180 قرار میدهیم.

برای عکس دوم: بازه ی **hue** برای **Pink.jpg** را $[150, 180] \cup [0, 12]$ در نظر گرفتیم. زیرا همانطور که میبینیم رنگ قرمز از هر دو طرف طیف مقداری را به خود اختصاص میدهد. نیاز به ایجاد محدودیت

دیگری نیست چون همانطور که میبینیم اختلاف رنگبندی در تصویر بسیار زیاد است. اکنون hue مربوط به تمام پیکسل هایی که شرایط گفته شده را دارند و درون بازه قرار میگیرند را برابر با 110 قرار میدهیم.

در این بخش دو تابع پیاده سازی شده که ابتدا توضیحشان میدهیم.

تابع shift :

- ورودی ها: {
- (1) ماتریس دو بعدی img یا عکس تک چنله است (دارای یک چنل میباشد زیرا نمایش آن سیاه سفید است)
 - (2) عدد صحیح dx که مقدار جابجایی لازم در جهت عمودی یا سطر های ماتریس را تعیین میکند \updownarrow
 - (3) عدد صحیح dy که مقدار جابجایی لازم در جهت افقی یا ستون های ماتریس را تعیین میکند $\leftarrow \rightarrow$

- خروجی ها: {
- (1) ماتریسی که به اندازه ی dx در جهت عمودی و به اندازه ی dy در جهت افقی انتقال یافته به طوری که سایز آن تغییری نکرده باشد و قسمت های اضافی، دور ریخته شود و قسمت های خالی با درایه هایی همرنگ حاشیه ی عکس پر شده باشد

عملکرد: اگر مقدار dx مثبت باشد به معنای انتقال به بالا و اگر منفی باشد به معنای انتقال به پایین است، همچنین اگر مقدار dy مثبت باشد به معنای انتقال به چپ و اگر منفی باشد به معنای انتقال به راست است. برای انجام این کار ابتدا ارتفاع و پهنای عکس را به ترتیب در متغیر های hei و wid نگه میداریم. سپس حاشیه ای (شامل تعداد برابری سطر و ستون که به چهار طرف ماتریس اضافه میشوند) به ضخامت padWidth با دستوری np.pad به دور ماتریس اضافه میکنیم. مقدار ضخامت این حاشیه برابر با ماکزیمم قدر مطلق dx و dy است زیرا بیشترین مقدار جابجایی لازم است. مقدار درایه های اضافه شده در حاشیه برابر با مقدار درایه ی گوشه بالا سمت چپ ماتریس یعنی [0, 0] است. زیرا فرض میشود که رنگ گوشه ترین پیکسل تصویر، مناسب ترین رنگ برای padColor حاشیه است. در تصویر داده شده در این تمرین، حاشیه سفید است و با توجه به 16 بیتی بودن عکس، حاشیه ای سفید رنگ، با مقدار 65535 به دور عکس اضافه میشود. مقدار dx و dy را با padWidth جمع میکنیم تا از حالت منفی خارج شوند و سپس عکس را از ارتفاع dx تا dx+hei و از پهنای dy تا dy+wid برش میدهیم و عکس حاصل را بعنوان خروجی بازگشت میدهیم. عکس نهایی عکسی است که به اندازه ی dx در جهت ارتفاع و به اندازه ی dy در جهت پهنای عکس جابجا شده بدون آنکه تغییری در ابعاد عکس بوجود آید زیرا بخش های خالی را با پیکسل های همرنگ با حاشیه ی عکس پر میکند و بخش های اضافی را دور میریزد. پس ارتفاع و پهنای عکس خروجی همان hei و wid است.

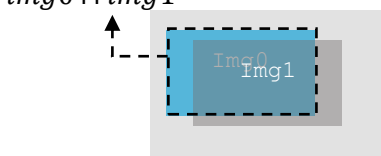
تابع match :

- ورودی ها:
- (1) ماتریس `img0` چنلی است که ثابت در نظر گرفته میشود
 - (2) ماتریس `img1` چنلی است که حرکت داده میشود تا بهترین انطباق صورت پذیرد
 - (3) ماتریس `img2` چنلی است که حرکت داده میشود تا بهترین انطباق صورت پذیرد
 - (4) عدد صحیح `sr` (searching range) که محدوده ی جستجو برای یافتن بهترین انطباق را تعیین میکند

- خروجی ها:
- (1) عدد صحیح `bestdx1` که بهترین مقدار جابجایی لازم برای چنل `img1` در راستای عمودی عکس را تعیین میکند
 - (2) عدد صحیح `bestdy1` که بهترین مقدار جابجایی لازم برای چنل `img1` در راستای افقی عکس را تعیین میکند
 - (3) عدد صحیح `bestdx2` که بهترین مقدار جابجایی لازم برای چنل `img2` در راستای عمودی عکس را تعیین میکند
 - (4) عدد صحیح `bestdy2` که بهترین مقدار جابجایی لازم برای چنل `img2` در راستای افقی عکس را تعیین میکند

عملکرد: هدف این است که با ثابت نگه داشتن چنل `img0`، دو چنل `img1` و `img2` را در محدوده ی جستجوی داده شده حرکت دهیم تا بهترین انطباق رخ دهد و مقدار جابجایی هر یک از این دو چنل تا رسیدن به بهترین انطباق در دو راستای عمودی و افقی را بعنوان خروجی بازگشت دهیم. اگر مقادیر `bestdx` ها مثبت باشند به معنای انتقال به بالا و اگر منفی باشد به معنای انتقال به پایین است، همچنین اگر مقادیر `bestdy` ها مثبت باشد به معنای انتقال به چپ و اگر منفی باشد به معنای انتقال به راست است. برای انجام این کار ابتدا ارتفاع و پهنای عکس ساکن را به ترتیب در متغیرهای `hei0` و `wid0` نگه میداریم. سپس حاشیه ای (شامل تعداد برابری سطر و ستون که به چهار طرف ماتریس اضافه میشوند) به ضخامت `padWidth` به دور ماتریس اضافه میکنیم. مقدار ضخامت این حاشیه برابر با همان محدوده ی جستجو `sr` (searching range) است تا اندیس های منفی بر روی این حاشیه قرار گیرند و معنا دار باشند. مقدار درایه های اضافه شده در حاشیه برابر با مقدار درایه ی گوشه بالا سمت چپ ماتریس یعنی `[0, 0]` است. زیرا فرض میشود که رنگ گوشه ترین پیکسل تصویر، مناسب ترین رنگ برای `padColor` حاشیه است. در تصویر داده شده در این تمرین، حاشیه سفید است و با توجه به 16 بیتی بودن عکس، حاشیه ای سفید رنگ، با مقدار 65535 به دور عکس اضافه میشود. معیار ما برای تشخیص بهترین انطباق، مجموع درایه های ماتریس مجذور تفاضلات نظیر به نظیر درایه های چنل ساکن و چنل متحرک است که در متغیر `dist` نگه داری میشوند (`dist1` معیاری برای انطباق

چنل $img1$ بر روی چنل $img0$ و $dist2$ معیاری برای انطباق چنل $img2$ بر روی چنل $img0$ (است).

$$dist1 = \sum_{i=0}^{hei0} \sum_{j=0}^{wid0} (img0_{i,j} - img1_{i,j}^{cropped})^2$$


با استفاده از دو `for` تو در تو در دو برابر محدوده ی جستوجو، هر یک از دو چنل متحرک را روی چنل ساکن دو دو راستای عمودی و افقی حرکت میدهم و در هر مرحله چک میکنیم که آیا معیار $dist$ برای هر چنل کمتر میشود یا بیشتر. اگر این معیار کمتر شود، یعنی به تطابق بهتری نسبت به مراحل قبل دست یافته ایم و اگر بیشتر شود یعنی به تطابق کمتری دست یافته ایم که این حالت مطلوب ما نیست. پس کمترین مقدار $dist$ به ما بهترین انطباق را نشان میدهد که آن را در متغیر min نگه میداریم. برای بررسی این معیار $dist$ در هر مرحله محاسبه میشود و با مقدار min که همان بهترین معیار بدست آمده از مراحل ماقبل است، مقایسه میشود. در ابتدای کار مقدار min ها را برابر با بینهایت در نظر میگیریم. در هر بار جابجایی اگر مقدار $dist$ کمتر از min شود، اطلاعات این جابجایی را که شامل جابجایی در راستای عمودی و افقی عکس میشود را در متغیر های $bestdx1$ ، $bestdy1$ ، $bestdx2$ و $bestdy2$ نگه میداریم و در نهایت پس از اتمام بررسی کمترین مجموع مجذور تفاضلات در تمام محدوده ی جستوجو، این چهار متغیر را بعنوان خروجی بازگشت میدهم. هر چه محدوده ی جستوجو را بزرگتر در نظر بگیریم، الگوریتم حالت های بیشتری را بررسی میکند اما زمان اجرا طولانی تر میشود. از آنجایی که ما میدانیم بهترین انطباق با اندکی جابجایی به دست میاید پس نیازی نیست که کل چنل متحرک را بر روی سرتاسر چنل ساکن حرکت دهیم. بلکه تنها در محدوده sb داده شده این بررسی را انجام میدهم.

حال پس از توضیحات مربوط به توابع پیاده سازی شده به توضیحات روند استفاده از آنها و رسیدن به خروجی مطلوب میپردازیم.

برای تکه کردن عکس ها به ابعاد برابر، ارتفاع عکس `melons.tif` را تقسیم بر سه میکنیم و به صورت عدد صحیح آن را در متغیر $height$ نگه میداریم. هنگام برش دادن عکس اصلی، پیکسل های دارای ارتفاع 0 تا $height$ را در $I0$ و پیکسل های دارای ارتفاع $height$ تا $2 \times height$ را در $I1$ و پیکسل های دارای ارتفاع $2 \times height$ تا $3 \times height$ را در $I2$ قرار میدهم. پس یعنی عکس اول و دوم و سوم را به ترتیب در متغیرهای $I0$ ، $I1$ و $I2$ نگه میداریم. به این ترتیب ارتفاع هر یک از سه قطعه عکس با یکدیگر مساوی و برابر با $height$ خواهد بود. همچنین پهنای

عکس ها نیز همگی با یکدیگر مساوی و برابر با پهنای عکس اولیه خواهد شد. در نتیجه هر سه عکس از نظر ابعاد با یکدیگر مساوی اند. عکس I_0 چنل ساکن ما است و دو عکس دیگر I_1 و I_2 چنل های متحرک اند. سه عکس به دست آمده کیفیت و تعداد پیکسل های زیادی دارند. در نتیجه محدوده ی جستجو که باید بعنوان ورودی به تابع match داده شود دارای پیکسل های زیادی است و برای بررسی انطباق آنها زمان زیادی صرف اجرای برنامه میشود. بنابراین با استفاده از روش هرمی سازی سرعت اجرا را افزایش میدهم. روش هرمی به این صورت است که کیفیت عکس را تا مقداری دلخواه میکاهیم و تابع match را روی تصویر کم کیفیت اثر میدهم و مقدار جابجایی لازم را میابیم و با استفاده از این مقادیر، میتوانیم مقدار جابجایی لازم برای عکس با کیفیت را به صورت حدودی تخمین بزنیم. مقدار جابجایی ای که در هر مرحله با استفاده از تابع match بدست میاید را در چهار متغیر x_1, y_1, x_2, y_2 و نگه میداریم. به طوری که مقدار جابجایی لازم تا رسیدن به بهترین انطباق در راستای عمودی و افقی برای عکس I_1 را به ترتیب در x_1 و y_1 و مقدار جابجایی لازم تا رسیدن به بهترین انطباق در راستای عمودی و افقی برای عکس I_2 را به ترتیب در x_2 و y_2 نگه میداریم.

در این الگوریتم هرم ما به تعداد r (repeat) تا تکرار دارد که در هر مرحله از تکرار اندازه ی عکس بی کیفیت x برابر میشود تا زمانی که به کیفیت اصل عکس برسیم. در هر مرحله اندازه ی عکس را تقسیم بر rr (resizing ratio) میکنیم و در هر مرحله عملیات match روی آن انجام میشود و متناسب با خروجی داده شده که شامل متغیر های dx_1, dy_1, dx_2, dy_2 عکس مرحله بعد را به اندازه ی $x \times dx_1, x \times dy_1, x \times dx_2, x \times dy_2$ پیکسل shift میدهم.

در بالا متغیر هایی تعریف شدند که مقدارشان در این تمرین به صورت زیر است:

ضریب تغییر سایز عکس $x=2$ یعنی در مرحله کیفیت عکس 2 برابر میشود تا به کیفیت اصلی برسیم.

تعداد تکرار $r=8$ یعنی پس از 8 بار تکرار به کیفیت اصلی عکس میرسیم.

نسبت تغییر سایز rr که در شروع $rr=2^{*(8-1)}=128$ است و در هر مرحله 2 برابر میشود.

محدوده ی جستجو $sb=30$ یعنی در محدوده ی 30 پیکسل بالا و پایین و چپ و راست را بررسی میکنیم.

تمامی متغیر های بالا قابل تغییر اند و میتوان آن ها را تغییر داد تا مراحل و شکل هرم تغییر کند.

در شروع عکس با کیفیت 1/128 را در محدوده ی تعیین شده ی برابر با sb=30 پیکسل match میکنیم و dx و dy هایی بعنوان خروجی تحویل میگیریم. اکنون کیفیت عکس را دو برابر میکنیم. یعنی کیفیت 1/64 را بررسی میکنیم. در اینجا بجای اعمال تابع match در یک محدوده ی بزرگ، dx و dy مرحله ی قبل را به کار میگیریم و با توجه به اینکه کیفیت عکس دو برابر شده پس مقادیر جابجایی dx و dy ها نیز دو برابر میشوند و بصورت تقریبی دو عکس جدید با عمل shift به اندازه ی $2 \times x1, 2 \times y1, 2 \times x2, 2 \times y2$ بر روی هم منطبق میشوند (با اندکی خطا) برای بر طرف کردن خطا باید بار دیگر تابع match را برای عکس با کیفیت 1/64 را صدا بزنیم ولی با این تفاوت که نیاز نیست به اندازه ی محدوده ی جستجوی sb=30 (مانند مرحله ی اول) این کار را انجام دهیم. بلکه نهایتا به اندازه ی sb=2 محدوده را بررسی کنیم و به اندازه ی (2) range هر چهار جهت چک میشود و به دقیق ترین انطباق برای عکس با کیفیت فعلی میرسیم. دلیل آن هم این است که اگر عکس نیاز به جابجایی به بیش از 1 پیکسل داشت، در مرحله ی قبلی این نیاز تشخیص داده میشد. زیرا دقت در واقع دو برابر شده (زیرا کیفیت عکس دو برابر شده). مثلا اگر متغیر x را 5 در نظر می گرفتیم، در هر مرحله کیفیت عکس 5 برابر میشد و دقت نیز 5 برابر میشد و نیاز بود که از بعد از مرحله ی اول sb=5 باشد تا دقت کافی اعمال گردد و مقدار جابجایی در (5) range قرار می گرفت. پس در حالت کلی، به جز مرحله ی اول که sb یک ناحیه ی بزرگ بود، در مراحل بعدی sb=x خواهد بود. (یعنی برابر با ضریب تغییر سایز عکس)

برای هر r=8 مرحله ای که در هر مرحله عکس x=2 برابر میشود									
i	مرحله	1	2	3	4	5	6	7	8
rr	نسبت تغییر سایز	128	64	32	16	8	4	2	1
sb	محدوده جستجو	30	2	2	2	2	2	2	2

از آنجایی که اطلاعات مربوط به جابجایی مورد نیاز از match های صورت گرفته در مراحل قبلی، به کارمان میانند و در هر مرحله روی عکس با کیفیت جدید اعمال میشوند. پس نباید این اطلاعات را دور بریزیم. ما در هر مرحله، جابجایی مورد نیاز را به ضریبی از متغیر جابجایی کل اضافه میکنیم (جابجایی مورد نیاز در این مرحله را با dx_i و dy_i و جابجایی کل از ابتدا تا کنون را با x_i و y_i نمایش میدهم) که در هر مرحله جابجایی مورد نیاز کل از مجموع جابجایی مورد نیاز همین مرحله و جابجایی کل مرحله ی قبل با وزن x برابر بدست میاید. (در اینجا x=2 است و با x برابر شدن عکس طی مرحله ی اخیر، معادلا جابجایی مورد نیاز این مرحله، x برابر جابجایی مورد نیاز تا مرحله ی ماقبل است)

$$\left. \begin{aligned} x1 &= x1 \times x + dx1 \\ y1 &= y1 \times x + dy1 \\ x2 &= x2 \times x + dx2 \\ y2 &= y2 \times x + dy2 \end{aligned} \right\}$$

عکس Dark.jpg را در متغیر I1 و عکس Pink.jpg را در متغیر I2 نگه میداریم. عکس I2 را هم اندازه ی عکس I1 میکنیم تا تعداد پیکسل هایشان یکسان شوند و در بررسی هیستوگرام ها به مشکل بر نخوریم. سپس عکس ها را به مختصات HSV میبریم (میتوان در مختصات BGR هم انجام داد) و سپس با for زدن برای هر چنل عملیات زیر را انجام میدهم. ابتدا هیستوگرام مربوط به دو عکس I1 و I2 را به ترتیب در hist1 و hist2 نگه میداریم. سپس نمودار تجمعی هر یک را حساب کرده و در cumHist1 و cumHist2 نگه میداریم. چنلی که در هر مرحله میخواهیم تغییر دهیم را در متغیر chnl و چنل هدف که مطلوب سوال است را در chnlTarget نگه میداریم. از آنجایی که توابع تجمعی صعودی اند و ما میخواهیم در هر نقطه $h_1(r) = h_2(s)$ یعنی $\text{cumHist1}[i1] = \text{cumHist2}[i2]$ به ازای هر i1 و i2 (که به ترتیب intensity های عکس I1 و I2 را نشان میدهند و در بازه ی [0,255] قرار دارند) برقرار باشد. با استفاده از دو for تو در تو به ازای هر i1، به دنبال i2 ای میگردیم تا فراوانی i1 و i2 در هیستوگرام تجمعیشان نزدیک ترین حالت ممکن باشد. برای این کار در هر مرحله قدرمطلق تفاضل فراوانی i1 و i2 را در هیستوگرام تجمعیشان بدست میآوریم و در متغیر dist نگه میداریم. اگر این مقدار از مراحل قبل کمتر شد آن را در متغیر min نگه میداریم و اطلاعات مربوط به بهترین انطباق نمودار ها را که شامل بهترین i2 ممکن برای متناظر کردن است را در besti2 نگه میداریم. اگر مقدار dist از مرحله ی قبلی بیشتر شد میدانیم که از مقدار مناسب داریم فاصله میگیریم (زیرا نمودار های تجمعی صعودی اند) پس break را صدا میزنیم. سپس به ازای تمام اندیس های دارای مقدار i1 در ماتریس chnl، متناظر ماتریس chnlTarget را با مقدار besti2 پر میکنیم. نهایتا chnlTarget را در چنل مربوطه اش در عکس I1 قرار میدهم و بار دیگر هیستوگرام آن را رسم میکنیم تا انطباق آن را نمایش دهد و در آخر عکس را به مختصات BGR باز میگردانیم. عکس I1 به روز رسانی شده به صورتی که هیستوگرام آن شبیه به عکس گل است. (رنگ **آبی** هیستوگرام عکس Dark.jpg قبل از تغییر و رنگ **قرمز** هیستوگرام عکس Pink.jpg و رنگ **سبز** هیستوگرام عکس Dark.jpg بعد از تغییر را نشان میدهد. سه نمودار بالا هیستوگرام عادی و سه نمودار پایین هیستوگرام تجمعی را نشان میدهند. با توجه به این که نمودار های زیر خواسته ی سوال نیستند از کد حذف شده اند و تنها سه نمودار سبز رنگ در پلات بالایی بعنوان خروجی ذخیره شده اند)

