فصل اول:

مقدمه

۱-۱- مقدمه

امروزه کشف و کشف چهره دارای کاربردهای متعددی از جمله در نظارت، بانکداری، تجهیزات چند رسانه‌ای مثل دوربین‌های عکاسی و کنسول‌های بازی رایانه ای و ... است. بسیاری از دوربین‌های دیجیتال جدید از کشف چهره برای متمرکز سازی خودکار استفاده می‌کنند. بعضی شرکت‌ها حتی از این هم فراتر رفته و از کشف چهره و سپس تحلیل آن برای کشف لبخند استفاده می‌کنند، به گونه‌ای که دوربین تنها زمانی عکس می گیرد که فرد لبخند بزند.

علاوه بر این اکثر مصرف کننده‌های دستگاه‌های الکترونیکی مانند موبایل، لپ تاپ، کنسول‌های و حتی تلویزیون‌ها شامل یک دوربین کوچک با محدوده وسیعی از ویژگی‌های پردازش تصویر از جمله کشف و شناسایی چهره هستند. به عنوان مثال تولید کنندگان تلویزیون‌های امروزی با نصب یک دوربین در برخی از سری تلویزیون‌ها قابلیت جدید کشف حضور کابر را امکان پذیر کرده‌اند. بطوریکه با کشف حضور کاربر و حتی سن کاربر می تواند برنامه های خاصی را که از پیش تعریف شده اند را غیر فعال و یا تلویزیون را برای صرفه‌جویی بیشتر خاموش کند.

از سوی دیگر، دیگر برنامه‌های درخواستی برای کشف و شناسایی چهره در دسته‌بندی اتوماتیک ویدئوها برای مقالبه با افزایش ذخیره سازی داده‌های دیجیتال استفاده می‌شوند. برای مثال دسته بندی محتویات پایگاه داده تلویزیون به وسیله برچسب زدن اتوماتیک برای تمام ویدئوهای که فرد

خاصی حضور دارد. در روشی مشابه تکنیک‌های کشف و شناسایی چهره به کاربران، موتورهای جستجو و برنامه‌های سازمان‌دهی تصاویر در جستجو خودکار تصاویر چهره کمک می‌کنند.

بعد از گذشت چهار دهه تحقیق و با طیف گسترده نرم‌افزارهای امروزی و امکانات جدید محققان هنوز در تلاش برای پیدا کردن الگوریتمی هستند که بهترین عملکرد را در محیط‌ها و نورهای مختلف و در تمامی زمان‌ها و کمترین خطا انجام دهد.

هدف از این تحقیق بررسی کارایی توصیف‌گر بافت ارائه شده توسط محققان دانشگاه Oulu در فنلاند برای کشف چهره است. این توصیف‌گر بافت تصاویر الگوی باینری محلی (LBP) نام دارد و مهمترین ویژگی آن سادگی و مقاوم بودن در برابر تغییرات روشنایی و وضعیت چهره است. از الگوی باینری محلی در ابتدا برای توصیف بافت‌های معمولی که در آن‌ها ارتباط مکانی پیکسل‌ها به اندازه تصاویر چهره نیست استفاده شده است. یک چهره می تواند به عنوان ترکیبی از میکروبافت که به وضیت محلی وابسته‌اند تصور کرد.الگوهای باینری محلی اساساً به دو توصیفگر مختلف تقسیم می شود: یکی سراسری و دیگری محلی. توصیفگر سراسری بیشتر برای تفکیک بلوک‌های غیر صورت استفاده می شود در حالی که از توصیفگر محلی به خاطر اطلاعات دقیق و خاصی که از صورت برای کشف و شناسایی چهره فراهم می‌کند استفاده می‌شود.

بنابراین روش ارائه شده شامل یک توصیف سراسری و همچنین یک توصیف محلی است که با تقسیم تصویر به چندین بلوک، محاسبه هیستورگام الگوهای باینری محلی برای هر بلوک و کنارهم قرار دادن این هیستوگرام‌ها بدست می آید. بافت نواحی مختلف چهره توسط الگوهای باینری محلی کدگذاری می شود در حالی که شکل نهایی با ترکیب هیستوگرام‌ها که بردار ویژگی نهایی را تشکیل می دهد بازیابی می گردد. در نهایت از این بردار ویژگی برای تغذیه یک طبقه‌بند مناسب که در اینجا ماشین بردار پشتیبان است استفاده می‌گردد تا در مورد چهره بودن یا نبودن تصویر داده شده تصمیم‌گیری شود.

این نگارش شامل ۵ فصل می‌باشد که فصل اول مقدمه ای درباره شبیه‌سازی و اطلاعاتی در مورد تصویر و بهینه سازی تصویر در برنامه متلب می‌باشد، فصل دوم کلیتی در مورد الگوی باینری محلی خواهد بود،در فصل سوم طبقه‌بند این شبیه‌سازی توضیح و توصیف شده است، در فصل چهارم طرز استفاده از الگوی باینری محلی و ماشین بردار پشتیبان تشریح شده است و در پایان کدهای شبیه-سازی و نکاتی در مورد نرم‌افزار متلب آمده است.

۱-۲- مفاهیم اولیه در پردازش تصویر

۱-۲-۱- مفهوم پیکسل در یک تصویر

پیکسل (pixel) شکل مختصر Picture Elements نقطه های بسیار ریز مربع شکلی هستند که از تجمع آنها، تصویر روی صفحه نمایش یا روی کاغذ (توسط چاپگر) شکل می گیرد. همان طور که بیت، کوچکترین واحد اطلاعات قابل پردازش توسط کامپیوتر است، پیکسل نیز کوچکترین عنصر سخت افزار و نرم افزار نمایشی یا چاپی است که برای شکل گرفتن تصاویر مورد استفاده قرار می‌گیرد. اگر برای هر پیکسل تنها دو رنگ (معمولا سیاه و سفید) در نظر گرفته شود، توسط یک بیت از اطلاعات قابل کددهی است و در صورتی که بیش از دو بیت برای ارائه یک پیکسل استفاده شود، محدوده رنگ ها یا سایه های خاکستری وسیع‌تری، قابل ارائه خواهد بود.

۱-۲-۲- دقت تصویر

دقت تصویر بستگی به شماره پیکسل‌ها دارد. با یک تصویر ۲ بیتی، حداکثر دامنه روشنایی ۲۲ یعنی ۴ می باشد که دامنه ان از ۰ تا ۳ تغییر می کند. در این حالت تصویر دقت (تفکیک پذیری لازم) را ندارد. تصویر ۸ بیتی حداکثر دامنه ۲۵۶ دارد و تغییرات ان بین ۰ تا ۲۵۵ است،که دقت بالاتری دارد. خوشبختانه در حال حاضر تکنیک‌هایی برای انجام این کار وجود دارد. از بین بردن نویزها به‌صورت نرمال توسط تعدادی از توابع ریاضی یا الگوریتم‌هایی که تحت عنوان 'treshholding' یا

'quantizing'نامیده می‌شود انجام می‌گردد. این فرایند بسیار حرفه‌ای و پیچیده‌ای است و نیاز به دانش و پشتوانه بالای ریاضی دارد. زمانی که خرابی‌ها از بین رفت، می‌توانیم پردازش عکس‌ها را ادامه دهیم که این کار با استخراج صورت‌ها و حالت‌ها از یک تصویر انجام می‌شود.

۱-۳- تصاویر دیجیتالی

یک تصویر را می توان توسط تابع دو بعدی f(x,y) که در آن x و y را مختصات مکانی و مقدار f در هر نقطه را شدت روشنایی تصویر در آن نقطه می نامند. اصطلاح سطح خاکستری نیز به شدت روشنایی تصاویر مونوکروم اطلاق می شود. تصاویر رنگی نیز از تعدادی تصویر دو بعدی تشکیل می‌شود.

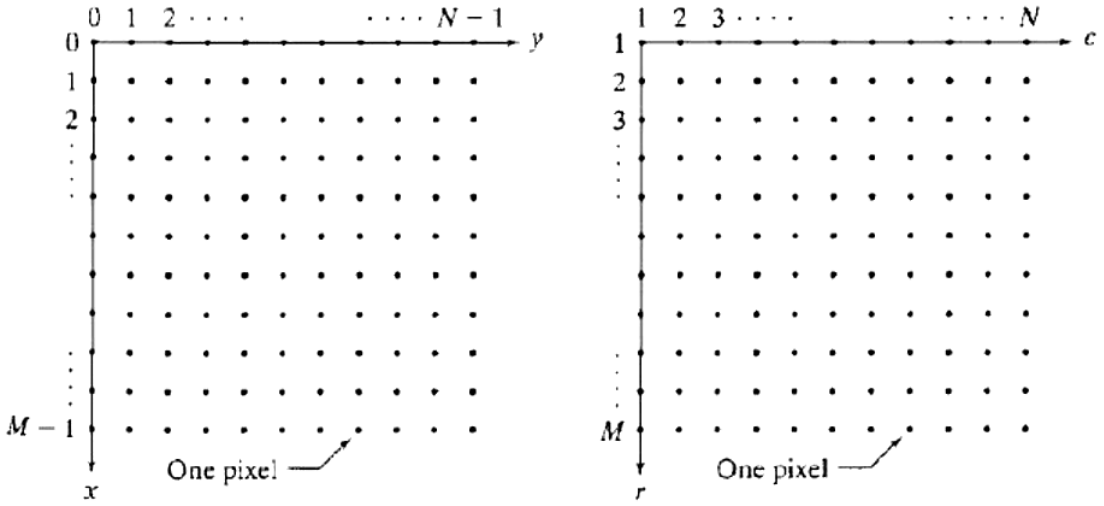
زمانی که مقادیر x و y و مقدار f(x,y) با مقادیر گسسته و محدود بیان شود، تصویر را یک تصویر دیجیتالی می نامند. دیجیتال کردن مقادیر x و y را sampling و دیجیتالی کردن مقدار f(x,y) را quantization گویند. یک تصویر دیجیتالی از تعدادی عناصر محدود با مقدار و موقیت مختلف تشکیل شده است. این عناصر، عناصر تصویر (picture elemnt) یا پیکسل (pixel) نامیده می شوند. پردازش تصویر دیجیتالی به معنی اعمال پردازش‌های مختلف روی یک تصویر دیجیتالی با استفاده از کامپیوتر دیجیتالی است.

۱-۳-۱- نمایش تصویر دیجیتالی

برای نمایش یک تصویر $m \times n$ از یک آرایه دو بعدی (ماتریس) که m سطر و n ستون دارد استفاده می کنیم. مقدار هر عنصر از آرایه نشان دهنده شدت روشنایی تصویر در ان نقطه است. در تمام توابعی که پیاده سازی خواهیم کرد، هر عنصر آرایه یک مقدار ۸ بیتی است که می تواند مقداری بین

۰ و ۲۵۵ داشته باشد. مقدار صفر نشان دهنده رنگ تیره (سیاه) و مقدار ۲۵۵ نشان دهنده رنگ روشن (سفید) است.

در بسیاری از کتاب‌های پردازش تصویر، مبدا به‌صورت زیر تعریف می شود: (x,y)=(0,0) مولفه x شماره سطر و مولفه y شماره ستون یک پیکسل را نشان می دهند. به‌دلیل اینکه در متلب اندیس مولفه آرایه ها از ۰ شروع نمی‌شود، در این محیط مبدا به‌صورت زیر تعریف می گردد: (r,c)=(1,1)



شکل ۱-۱ معرفی پیکسل

۱-۴- انواع تصاویر

تصاویر در متلب شامل یک ماتریس داده می باشند و معمولا به ماتریس جعبه رنگ وابسته هستند. سه نوع ماتریس داده برای تصاویر وجود دارد که هر کدام به‌صورت های مختلفی تفسیر می شوند:

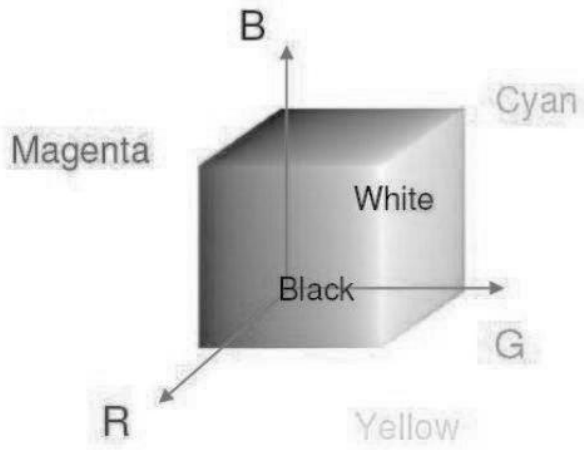
- تصاویر اندیس گذاری شده (indexed)
- تصاویر RGB
- تصاویر با شدت رنگ (intensity)

۱-۴-۱- تصاویر اندیس گذاری شده

در واقع یک تصویر اندیس گذاری شده آرایه ای دو بعدی است که در هر خانه ی آن شماره رنگ پیکسل مورد نظر ذخیره می شود. تصویر اندیس گذاری شده، به یک جعبه رنگ نیاز دارد و داده های تصویر را به‌صورت اندیس هایی در ماتریس جعبه رنگ تفسیر می کند. ماتریس جعبه رنگ، یک جعبه استاندارد است که اندازه آن 3*M بوده و شامل مقادیر قابل قبول RGB است. با ارائه ی آرایه ی داده ای x(i,j) که مربوط به تصویر است و آرایه ی جعبه رنگ cmap، رنگ هر پیکسل از تصویر با: cmap(x(i,j)) مشخص می شود که این رابطه نشانگر این است که مقادیر X، اعداد صحیحی در بازه [1 length(cmap)] است.

۱-۴-۲- تصاویر RGB

تصاویر رنگی، از ترکیب چند تصویر دو بعدی تشکیل شده اند. به طور مثال در سیستم رنگی RGB هر تصویر از سه مولفه تصویر قرمز، سبز و آبی تشکیل شده است.



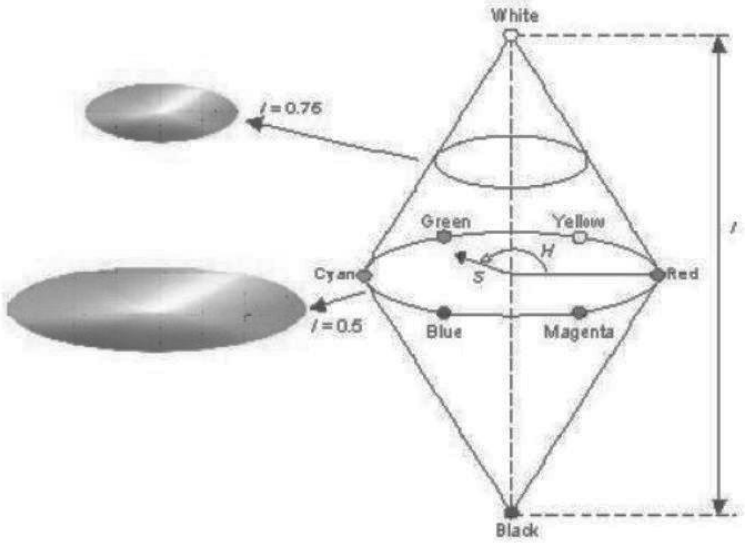
شکل ۱-۲ مکعب رنگ RGB

۱-۴-۳- تصاویر با شدت رنگ

همانطور که مشاهده کردیم در حالت عادی ۳ درجه رنگی از ۳ رنگ اصلی آبی و سبز و قرمز برای هر نقطه وجود دارد که با کم و زیاد شدن شدت هر مولفه اصلی، رنگ حاصل تغییر می کند. اما در پردازش تصویر این فرمت اصلا مناسب نیست، علت این است که در پردازش تصویر معمولا نیاز به کشف یک محدوده رنگی داریم. مثلا رنگ نارنجی، در این حالت یافتن این محدوده رنگی با ترکیب ۳ رنگ اصلی ممکن نیست با اینکه کاری بسیار دشوار و وقت گیر می باشد. این سیستم یا روش دیگری که در این زمینه وجود دارد بر اساس قاعده زیر است.

برای هر رنگ سه مشخصه وجود دارد

- نام رنگ (Hue): که مشخص کننده نوع رنگ می باشد. به عنوان مثال آبی کمرنگ یا قرمز متمایل به نارنجی یا ...
- شدت رنگ (Saturation): هر رنگ می تواند پر رنگ یا کمرنگ باشد اما ماهیت ذاتی آن ثابت است و فقط کم رنگ یا پر رنگ تر شده است.
- روشنایی یا تیرگی رنگ (Intensity): این مولفه شدت رنگ را تعیین می کند که با نور تابیده شده به آن تغییر می کند. به عنوان مثال اگر نور تابیده شده کم باشد، رنگ رو به تیرگی رفته و تقریبا متمایل به سیاه می شود. بنابراین حوضه جدیدی از رنگ که الهام گرفته از چشم انسان می باشد قابل تعریف است. به این حوضه رنگی اصطلاحا HSI گفته می شود که مخفف ۳ کلمه بالاست. شکل زیر نمودار تغییرات رنگ را با توجه به این ۳ مولفه بیان می کند:



شکل ۱-۳ مخروط نمودار تغییر رنگ

H: بین ۰ تا ۳۶۰ قابل تغییر است، یعنی از قرمز تا سبز ۱۲۰ و از سبز تا آبی ۱۲۰ و از آبی تا قرمز ۱۲۰ درجه در خلاف جهت عقربه های ساعت.

S: بین ۰ تا ۱۰۰ قابل تغییر است که از کم رنگ (0) تا پر رنگ (100)

I: از ۰ تا ۱۰۰ تغییر می کند، یعنی از تاریک (0) تا روشن (100)

چند نمونه از رنگ ها در ۲ حوزه مذکور:

رنگ	مقادیر RGB	مقادیر HIS
سیاه	(۲۵۵ و ۲۵۵ و ۲۵۵)	(۰ و ۰ و ۰)
سفید	(۲۵۵ و ۲۵۵ و ۲۵۵)	(۱۰۰ و ۰ و ۰)
قرمز	(۰ و ۰ و ۲۵۵)	(۱۰۰ و ۱۰۰ و ۰)
سبز	(۰ و ۲۵۵ و۰)	(۱۰۰ و ۱۰۰ و ۱۲۰)
آبی	(۲۵۵ و ۰ و ۰)	(۱۰۰ و ۱۰۰ و ۲۴۰)
قهوه ای	(۱۲۸ و ۱۲۸ و ۶۴)	(۵۰ و ۵۰ و ۱۸۰)

کار کردن با سیستم HSV نیز دقیقا معادل سیستم HSI می باشد. بنابراین به عنوان مثال با داشتن یک تصویر با فرمت HSV می توان به راحتی مکان نقاط با رنگ های مورد نظر را روی تصویر یافت.

فصل دوم:

الگوی باینری محلی

۲-۱- مقدمه

پس از گذشت ۴۰ سال، تجزیه و تحلیل چهره حوزه پژوهشی مهمی به‌دلیل طیف گسترده برنامه‌های کاربردی بوده است، مانند: اجرای قانون، نظارت، سرگرمی مانند بازی‌های ویدئویی و واقعیت مجازی، امنیت اطلاعات، بانکداری، واسط کامپیوتری و غیره.

در حال حاظر تشخیص و شناسایی چهره هنوز یک چالش بسیار دشواری است و هیچ روش منحصر به فردی وجود ندارد که راه حل قوی و کارآمدی برای همه شرایط پردازش چهره که ممکن است با آن رو به رو شود را فراهم کند. در برخی شرایط کنترل شده، تشخیص و شناسایی چهره تقریباً حل شده و یا حداقل در حال حاظر میزان دقت بالایی دارند اما در برخی دیگر برنامه‌های کاربردی که شرایط کنترل شده نیست تجزیه و تحلیل چهره هنوز هم نشان دهنده چالشی بزرگ است. در این بخش خلاصه‌ای از الگوریتم‌های اصلی همراه با چالشی‌ترین توصیفگرها برای تشخیص و شناسایی چهره ارائه شده است.

۲-۲- کشف چهره

در بیشتر موارد، در این زمینه پژوهش اینگونه فرض می‌شود که چهره در یک تصویر یا توالی ویدئو پیش از قبل شناخته و متمرکز شده است. بنابراین برای ساخت یک سیستم قوی یک روش کشف کارآمد و کاملاً اتوماتیک یک گام ضروری برای موفقیت در هر برنامه پردازش تصویر مورد نیاز است.

کشف چهره حالت خاصی از بحث کشف کلاس-شی است، که وظیفه اصلی آن پیدا کردن موقعیت و اندازه اجسام در تصویر مربوط به کلاس داده شده است. الگوریتم های کشف چهره در مرحله اول در کشف چهره انسان از رو به رو و بطور کامل متمرکز شده است. اما امروزه آن‌ها کوشش می کنند برای حل عمومی‌تر کشف چند طرفه: چرخش در صفحه و چرخش بیرون از صفحه. با این تفاسیر کشف چهره هنوز یک چالش بسیار مشکل با توجه به تنوع زیاد در اندازه، شکل، رنگ و بافت چهره انسانهاست.

بطور کلی الگوریتم های کشف چهره به عنوان مسئله طبقه‌بندی باینری پیاده سازی می شود. این بدان معناست که با دادن تصویر ورودی تصویر مورد نظر به بلوک‌های تقسیم شده و هر بلوک نیز به یک بردار ویژگی تبدیل می‌شود.از ویژگی‌های کلاس صورت و کلاس غیر صورت برای آموزش طبقه‌بند خاص استفاده می‌شود. سپس یک تصویر ورودی جدید داده می شود که طبقه‌بند قادر به کشف چهره بودن یا نبودن نمونه خواهد بود.

بطور کلی روش های کشف چهره به گروه‌های زیر دسته بندی شده است:

- روش‌های مبتنی بر آگاهی: این تکنیک پایه قوانین کد گذاری دانش و آگاهی بشر در مورد رابطه بین ویژگی‌های صورت می‌باشد([15]Yang).
- روش‌های مبتنی بر ویژگی‌های تغییرناپذیر: (به عنوان مثال ویژگی‌های صورت (Yow & Cipolla [16])، بافت و ویژگی‌های متعدد):
- روش‌های مبتنی بر الگو: (به عنوان مثال الگوهای از پیش تعیین شده و الگوهای دگردیس پذیر([17]Yuille)): این روش مبتنی بر استفاده استاندارد الگوی صورت است که می تواند به‌صورت دستی از پیش تعیین شده یا به وسیله یک تابع پارامتریزه شده باشد.
- روش‌های مبتنی بر نمود: (به عنوان مثال شبکه‌های عصبی ([18] Juell) ماشین بردار پشتیبان ([20] Schulze)، مدل مخفی مارکوف ([21] Rabiner) و Eigenfaces (Turk &

(Pentland [22]) بر خلاف روش‌های جستجو مدل، مدلهای مبتنی بر ظاهر و الگوها تولیدکننده آموزش مجموعه‌ای از تصاویر حاوی تغییرات نمایندگی کلاس‌های چهره هستند.

واضح است این دسته بندی‌ها بهم وابسته هستند و می توانند در جهت بهبود میزان کشف چهره با هم ترکیب شوند. بهتر است روش‌های انتخاب شود که وابستگی کمتری باهم‌دیگر داشته که این امر باعث نتیجه بهتر و هدر رفت زمانی کمتری خواهد شد.

به منظور مقایسه روش‌های مختلف، پارامترهای مختلف را می توان در سیستم‌های کشف چهره استفاده کرد. بطور معمول عملکرد با استفاده از نرخ کشف صحیح و نرخ کشف غلط می‌باشد. سپس خطاهای معمول در طرح‌های کشف چهره عبارتند از:

False negative: کشف نداده شدن چهره به علت نرخ کشف کم

False positive: کشف دادن اشتباه چهره به علت نرخ کشف غلط بالا

تجزیه و تحلیل این پروژه روش کشف چهره بر اساس الگوهای باینری محلی است که می توان به عنوان روش مبتنی بر نمود در نظر گرفت.

۲-۳- چالش

به منظور درک بهتر وظیفه و مشکلات تشخیص و شناسایی چهره عوامل زیر باید در نظر گرفته شوند، زیرا این عوامل می توانند سبب جدی کاهش کیفیت اجرا سیستم تشخیص و شناسایی چهره شوند.

- وضعیت: تصویر چهره به‌دلیل تغییرات موقعیت نسبی میان چهره و دوربین و برخی از ویژگی-های صورت مانند چشم‌ها و بینی می توانند تا حدی یا کاملاً پنهان شوند.

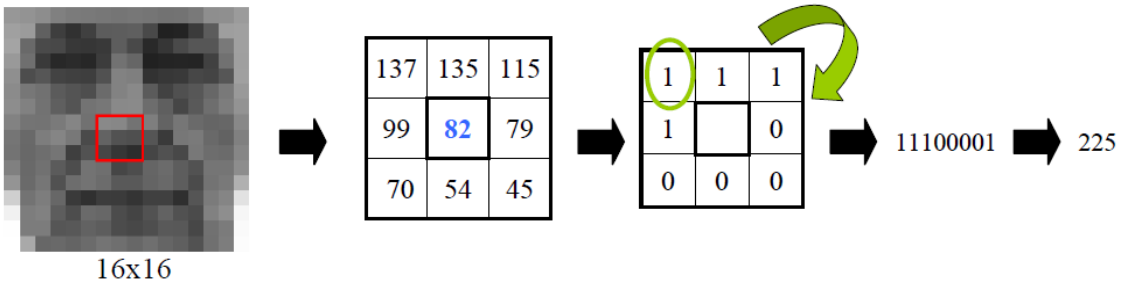
- نورپردازی و دیگر شرایط تصویر: سیما چهره در یک تصویر می تواند با عواملی مانند تغییرات زاویه نور و شدت آن و ویژگی‌های دوربین مانند واکنش سنسور و لنز تحت تاثیر قرار گیرد. تفاوت‌های ایجاد شده توسط این عوامل می توانند بیشتر از تفاوت‌های بین افراد باشد.
- انسداد: چهره به وسیله اشیا (ریش، سبیل و عینک) و حتی دیگر افراد ممکن است مسدود شود.
- حالات چهره: ظاهر چهره به طور مستقیم چهره فرد را تحت تاثیر قرار می دهد.

علاوه بر این عوامل زیر وظایف خاص از کشف چهره و مرتبط با کارایی محاسباتی هستند که خواستار صرف کوتاه‌ترین زمان در مناطق غیره چهره‌اند.

- تعداد چهره‌های در تصویر: به ندرت در تصاویر چهره یافت می‌شود. بین ۰ تا ۱۰ چهره در یک تصویر پیدا خواهد شد.
- اسکن مناطق: پنجره لغزنده ترکیب گسترده‌ای از مکان‌ها و مقیاس‌ها را ارزیابی می کند و مستقیماً وابسته به پارامترهای زیر است:
  - گام اسکن (برای اکتشاف محل)
  - Down-sampling ratio (برای کشف مقیاس)

۲-۴- توصیف بافت

الگوی باینری محلی در ابتدا به عنوان توصیف‌گر بافت عمومی معرفی شد. این عملگر با مقایسه همسایگی ۳×۳ هر پیکسل از یک تصویر با خود آن پیکسل و در نظر گرفتن نتیجه به‌صورت یک عدد دودویی، به آن پیکسل یک برچسب نسبت می‌دهد. در نشریات مختلف، مقادیر دایره ای به‌صورت ۰ و ۱ و در جهت عقربه های ساعت یا عکس جهت عقربه‌های ساعت خوانده می شود. در این تحقیق نتیجه به‌صورت عدد باینری خواهد بود که از بالا سمت چپ به‌صورت ساعت‌گرد بدست می‌آید که در شکل زیر نشان داده شده است.



شکل ۲-۱ نحوه بدست آوردن برچسب الگوی باینری محلی به‌صورت ساعت‌گرد

به بیان دیگر برای یک پیکسل دلخوه $LBP\ (x_c.y_c)$ به‌صورت مجموعه‌ای منظمی از مقایسه‌های مقادیر پیکسلی بین پیکسل مرکزی و پیکسل‌های همسایگی مقایسه می‌شوند. برچسب دهدهی نهایی نتیجه شده از این کلمه ۸ بیتی به‌صورت زیر بیان می‌شود.

$$LBP(X_c, Y_c) = \sum_{n=0}^{7} s(I_n - I_c)2^n$$

به این‌صورت که I_c مربوط به ارزش خاکستری پیکسل مرکزی (X_c,Y_c) و I_n ارزش خاکستری از ۸ پیکسل اطراف آن و تابع $s(k)$ به‌صورت تعریف می‌شود:

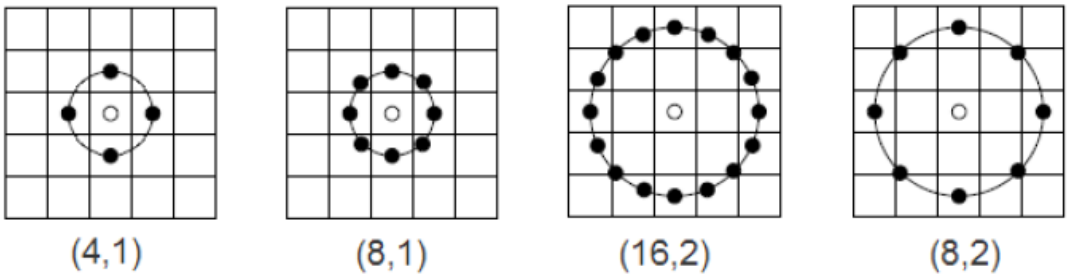
$$s(k) = \begin{cases} 1 & \text{if} \quad k \geq 0 \\ 0 & \text{if} \quad k < 0 \end{cases}$$

۲-۵- توسعه الگوی باینری محلی

عملگر الگوی باینری محلی برای بررسی بافت‌ها در مقیاس‌های مختلف گسترش داده شده به گونه‌ای که از همسایگی‌هایی با اندازه مختلف استفاده می کند. با استفاده از دایره همسایگی و درونیابی خطی از مقادیر پیکسل هر شعاع و تعداد نمونه‌ها در همسایگی می توان به کار گرفت. از این رو یاد داشت زیر تعریف شده است.

(P,R) به این معنیست P به معنی نقاط نمونه برداری با شعاع R است.

شکل زیر چند نمونه از نقاط نمونه برداری های مختلف و شعاع آن‌ها را نشان می دهد:



شکل ۲-۲ نقاط نمونه برداری و شعاع‌های مختلف

در مورد LBP(4.1) دلیل اینکه چرا ۴ نقطه عمودی و افقی انتخاب شده این است که چهره شامل لبه‌های افقی و عمودی بیشتری نسبت به قطرهاست.

در هنگام محاسبه عملیات پیکسل در همسایگی N×N در مرز تصویر، بخشی از این پوشش که از لبه تصویر بیرون است غیر فعال می گردد. در چنین موقعیت‌هایی تکنیک‌های پهن کردن مختلفی از قبیل پهن کردن تصویر به‌صورت صفر کردن و تکرار عناصر مرزی یا روش آینه ای استفاده می شود. با وجود این در مورد عملگر الگوی باینری محلی مرز بحرانی تعریف شده توسط شعاع R از عملگر دایره‌ای با استفاده از روش پهن کردن حل نمی شود. به جای آن عملگر از پیکسل (R.R) تصویر شروع به کار می کند. مزیت هیستوگرام نهایی الگوی باینری محلی این است که تحت تاثیر مرزهای نخواهد بود. اگر چه اندازه برچسب تصویر الگوی باینری محلی به اندازه (Width-R)x(Height-R) پیکسل کاهش پیدا خواهد کرد.

۲-۶- الگوهای یکنواخت

Ojala[7] در کار خود gary-scale چند رزولیشنی و چرخش یکسان طبقه بندی بافت با الگوهای باینری محلی، نشان می‌دهد که ممکن است برای تنها یک زیر مجموعه از 2^P الگوی باینری محلی برای توصیف بافت تصاویر استفاده شود. این مجموعه الگوهای یکنواخت یا الگوهای اساسی نامیده می

شود. یک الگوی باینری محلی زمانی یکنواخت گفته می‌شود که الگـوی دودویـی دوار (جهـت عقربـه های ساعت) شامل حداکثر ۲ انتقال از ۰ به ۱ و بالعکس باشد.

جدول ۲-۱ مثالی از الگوهای محلی یکنواخت و غیریکنواخت

Circular Binary Pattern	# of bitwise transitions	Uniform pattern?
11111111	0	Yes
00001111	1	Yes
01110000	2	Yes
11001110	3	No
11001001	4	No

. مثالی از الگوهای محلی یکنواخت و غیریکنواخت.

هر یک از این الگوها دارای ستون مختص به خود در هیستوگرام الگوی باینری محلـی هسـتند. بقیـه الگوهایی که بیش از ۲ انتقال دارند در یک تک ستون انباشته می شوند. با توجه به تجربه ما الگوهای غیریکنواخت در ستون ۰ انباشه خواهد شد.

این نوع از هیستوگرام الگوی باینری محلی به‌صورت $LBP^{u2}_{(P,R)}$ نشان داده شده و حاوی کمتـر از 2^P ستون است. نظر به داشتن هیستوگرام الگوی باینری محلی بـه‌صـورت P=8 و u^2، ۸ سـتون برچسـب باینری با تنها ۵۸ ارزش یکنواخت از یک مجموعه ۲۵۶ تایی مختلف بدست می‌آید. با توجه به این که یک ستون به منظور نشان دادن الگوهای غیر یکنواخت مورد نیاز است، یـک هسـیتوگرام ۵۹ سـتونه هیستوگرام پایانی $LBP^{u2}_{(P,R)}$ خواهد بود. در نتیجه کاهش ٪۷۶,۹۵ در بردار ویژگـی بدسـت خواهـد آمد. این کاهش به‌دلیل کافی بودن الگوهای یکنواخت برای توصیف بافت یک تصویر هستند، به عنوان مثال Ojala [7] در پژوهش خود به این موضوع اشاره کرده و همچنین می توانیم در سه جـدول زیـر کل الگوی یکنواخت حاظر در موارد مختلف بافت تصاویر را مشاهده کنیم.

با یک نگاه، می توان به وضوح مشاهده کرد که بافت تصاویر به طور عمده توسط الگوهـای یکنواخـت تشکیل شده‌اند (~ ٪۸۰)، بنابراین در هیستوگرام الگوی باینری محلی منطقـی اسـت بـه هـر الگـوی

یکنواخت یک ستون جداگانه اختصاص بدهیم و بـه تمـام الگوهـای غیـر یکنواخـت نیـز یـک سـتون اختصاص دهیم.

مقادیر زیر برای $LBP^{u2}_{(P,R)}$ از تصاویر ۱۶×۱۶ استخراج شده که از اینترنت جمع آوری شده‌اند.

جدول ۲-۲ درصد الگوی یکنواخت برای $LBP^{u2}_{(P,R)}$ با استفاده از تصاویر ۱۶×۱۶

16x16 Images	Number of samples	% Uniform patterns in the image
Faces	6650	81,29 %
Non Faces	4444432	82,49 %

نتایج فوق بسیار شبیه به نتایج بدست آمده توسط Ojala[7] در آزمایشاتش با بافت تصاویر می باشد.

جدول ۲-۳ نتایج بدست آمده توسط Ojala: درصد الگوی یکنواخت برای موارد مختلف LBP

LBP case	% Uniform patterns in the image
(8, 1)	90%
(16,2)	70%

و همچنین شبیه نتایج T.Ahonen[1] با استفاده از تصاویر ۱۹×۱۹ پایگاه داده FERET.

جدول ۲-۴ نتایج بدست آمده توسط T.Ahonen [1]: درصد الگوی یکنواخت برای موارد مختلف LBP با پایگاه داده FERET

LBP case	% Uniform patterns in the image
(8, 1)	90.6%
(8,2)	85.2%

مثال زیر برای روشن شدن روش محاسبه هیستوگرام الگوی باینری محلی به وسیله جزئیات استخراج هیستوگرام $LBP^{u2}_{(P,R)}$ از یک تصویر چهره ۱۶×۱۶ در نظر گرفته شده است.

۱. برای هر پیکسل در تصویر عملگر $LBP_{(8.1)}$ استخراج شده اسـت. توجـه کنیـد کـه مرزهـای تصویر نادیده گرفته می شوند با توجه به این واقعیت کـه همسـایگی ۳×۳ مـورد نیـاز اسـت. بنابراین نتیجه برچسب تصویر ۱۴×۱۴ $LBP_{(8.1)}$ بدست می‌آید. مشاهده می کنید که مقادیر

سیاه (۰ خاکستری) متناظر با برچسب غیریکنواخت و مقادیر سفید (۲۵۵ سطح خاکستری) به برچسب انتقال ۰ بیتی ('۱۱۱۱۱۱۱۱' یا '۰۰۰۰۰۰۰۰') مطابقت دارد.

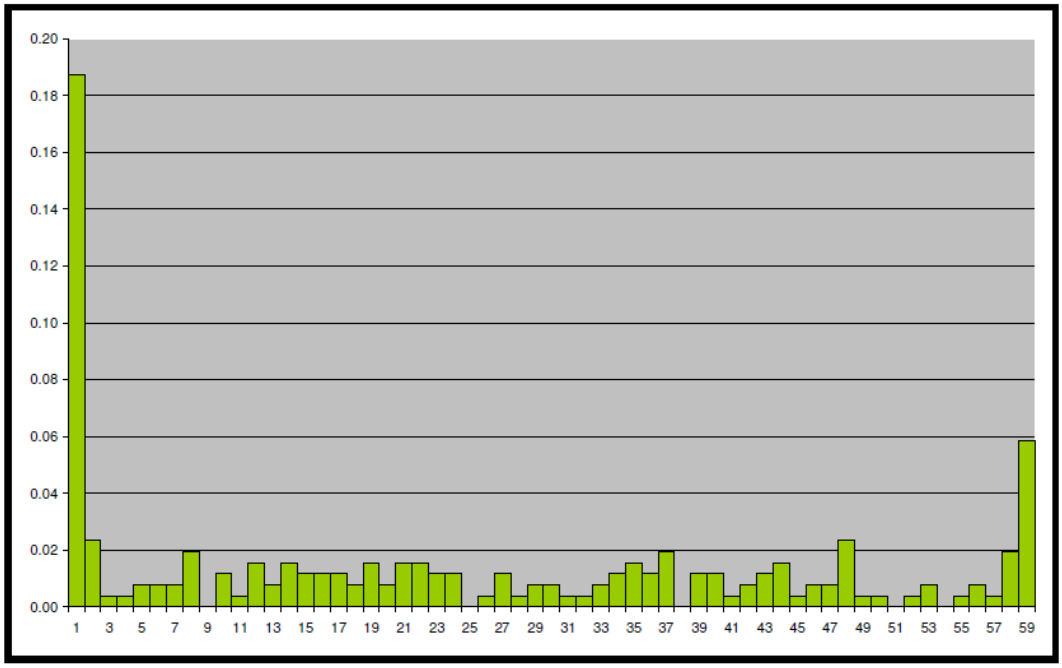
شکل ۲-۳ $LBP_{(8.1)}$ استخراج شده از نتایج تصاویر چهره در برچسب تصویر ۱۴×۱۴

۲. سپس ۵۹ ستون $LBP^{u2}_{(P,R)}$ توسط جمع تمام الگوهای غیریکنواخت با بیش از ۲ انتقال به ستون ۰ محاسبه می شود، پس از آن، بقیه الگوهای یکنواخت در یک ستون اختصاصی انباشته می شوند. به شکل زیر برای مرور کلی چگونگی الگوهای یکنواخت در برچسب هیستوگرام رجوع کنید.

جدول ۲-۵ برچسب LBP ستون مربوط به هسیتوگرام نهایی $LBP^{u2}_{(P,R)}$

Uniform Label (decimal)	Uniform Label (binary)	Number of Transitions	$LBP^{u2}_{(8,R)}$ histogram bin
non uniform patterns	non uniform patterns	>2	0
0	00000000	0	1
1	00000001	1	2
2	00000010	2	3
3	00000011	1	4
4	00000100	2	5
⋮	⋮	⋮	⋮
251	11111011	2	54
252	11111100	1	55
253	11111101	2	56
254	11111110	1	57
255	11111111	0	58

شکل بعدی ۵۹ ستون هیستوگرام $LBP^{u2}_{(P,R)}$ ، که در آن بالاترین انتخـاب (سـتون ۰ از اول) کـه مربوط به الگوهای غیریکنواخت ۱۸,۷۵٪ از مقادیر کل برچسب های ۱۴×۱۴ است نشان می‌دهد. در نتیجه، ۸۱,۲۵٪ برای مقادیر یکنواخت است. علاوه بر این، جالب است توجه کنید که انتخاب مهم دوم در هیستوگرام (ستون ۵۸ از آخر) ۵,۸۵٪ از برچسب های هیستوگرام است و مربوط به ۲۵۵ برچسب با ۰ انتقال بیتی ('۱۱۱۱۱۱۱۱') می باشد.



شکل۲-۴ ۵۹ ستون هیستوگرام $LBP^{u2}_{(P,R)}$ از تصاویر چهره ۱۴×۱۴

فصل سوم:

طبقه بندی ماشین بردار پشتیبانی(SVM)

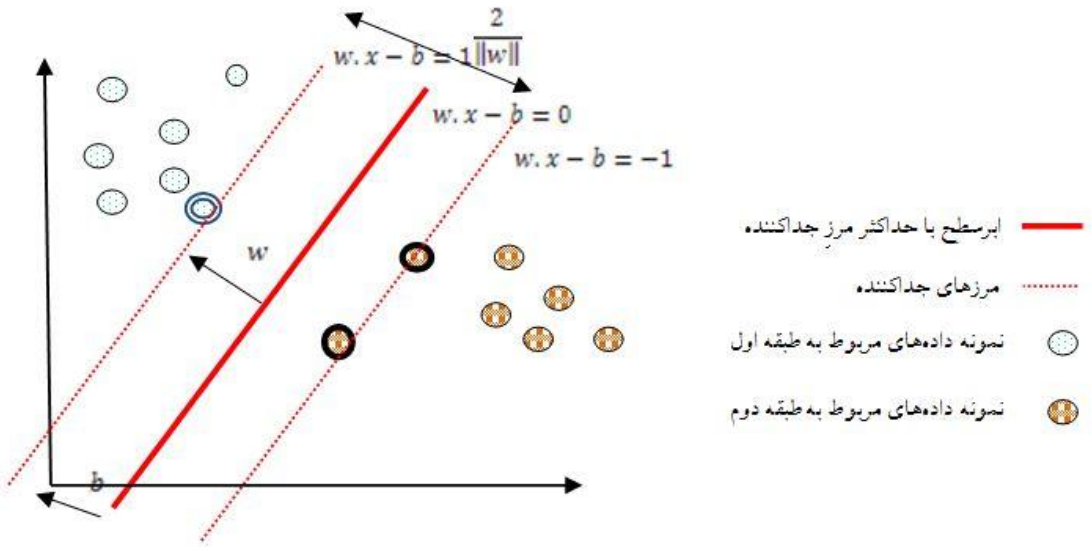
۳-۱- مقدمه

ماشین بردار پشتیبانی (Support vector machines – SVM) یکی از روش‌های یادگیری با نظارت است که از آن برای طبقه‌بندی و رگرسیون استفاده می کنند. این روش از جمله روش‌های نسبتاً جدیدی است که در سال‌های اخیر کارایی خوبی نسبت به روش‌های قدیمی‌تر برای طبقه‌بندی از جمله شبکه‌های عصبی پرسپترون نشان داده است. مبنای کار دسته‌بندی کننده ماشین بردار پشتیبان دسته‌بندی خطی داده است و در تقسیم خطی داده‌ها سعی می‌کنیم خطی را انتخاب کنیم که حاشیه اطمینان بیشتری داشته باشد. حل معادله‌ی پیدا کردن خط بهینه برای داده‌ها به وسیله روش‌های QP که روش‌های شناخته شده‌ای در حل مسائل محدودیت‌دار هستند صورت می‌گیرد. قبل از تقسیم خطی برای اینکه ماشین بتواند داده‌های با پیچیدگی بالا را دسته‌بندی کند داده‌ها را به وسیله تابع phi به فضای با ابعاد خیلی بالاتر می بریم. برای اینکه بتوانیم مساله ابعاد خیلی بالا را با استفاده از این روش‌ها حل کنیم از قضیه دوگانی لاگرانژ برای تبدیل مساله مینیمم سازی مورد نظر به فرم دوگانی آن که در آن بجای تابع پیچیده phi که ما را به فضایی با ابعاد بالا می‌برد، تابع ساده‌تری به نام تابع هسته که ضرب برداری تابع phi است ظاهر می‌شود استفاده می‌کنیم. از تابع هسته مختلفی از جمله هسته‌های نمایی، چند جمله‌ای و سیگموید میتوان استفاده نمود.

۳-۲- روش طبقه‌بندی ماشین بردار پشتیبان

فرض کنیم مجموعه نقاط داده $\{(x_1, c_1), (x_2, c_2), \dots, (x_n. c_n)\}$ را در اختیار داریم و می‌خواهیم آنها را به دو طبقه $c_i = \{-1,1\}$ تفکیک کنیم. هر x_i یک بردار p بعدی از اعداد حقیقی است که در واقع همان متغیرهای بیانگر رفتار نرم افزار هستند.

روشهای طبقه بندی خطی، سعی دارند که با ساختن یک ابرسطح (که عبارت است از یک معادله خطی)، داده‌ها را از هم تفکیک کنند. روش طبقه بندی ماشین بردار پشتیبان که یکی از روشهای طبقه‌بندی خطی است، بهترین ابرسطحی را پیدا می‌کند که با حداکثر فاصله (maximum margin)، داده‌های مربوط به دو طبقه را از هم تفکیک کند. به منظور درک بهتر مطلب، در شکل زیر تصویری از یک مجموعه داده متعلق به دو کلاس نشان داده شده که روش ماشین بردار پشتیبان بهترین ابرسطح را برای جداسازی آنها انتخاب می کند.



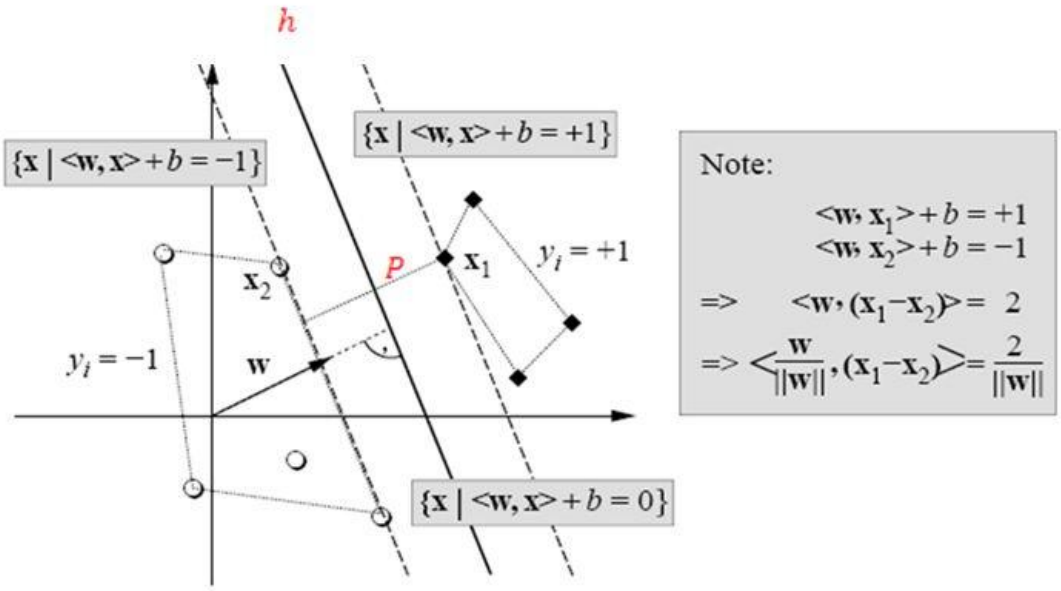
شکل ۳-۱ ابرسطح با حداکثر مرزِ جداکننده به همراه مرزهای جداکننده برایِ طبقه بندی نمونه

داده‌های مربوط به دو طبقه متفاوت. نمونه‌های قرار گرفته بر روی مرزها بردارهای پشتیبان نام دارند.

در این شکل داده‌ها دو بعدی هستند یعنی هر داده تنها از دو متغیر تشکیل شده است.

۳-۳- نحوه تشکیل ابرسطح جداکننده توسط ماشین بردار پشتیبان

در این بخش می خواهیم نحوه ساخت ابرسطح جداکننده را بر روی یک مثال با جزئیات شرح دهیم. تصویر دقیقی از نحوه تشکیل ابرسطح جداکننده توسط ماشین بردار پشتیبان در شکل زیر نشان داده شده است.



شکل ۳-۲ نحوه ساخت ابرسطح جداکننده بین دو طبقه داده در فضای دو بعدی

ابتدا یک convex (پوسته محدب) در اطراف نقاط هر کدام از کلاسها در نظر بگیرید. در شکل بالا در اطراف نقاط مربوط به کلاس -1 و نقاط مربوط به کلاس $+1$ پوسته محدب رسم شده است. خط P خطی است که نزدیکترین فاصله بین دو پوسته محدب را نشان می دهد. h که در واقع همان ابرسطح جداکننده است، خطی است که P را از وسط نصف کرده و بر آن عمود است.

b عرض از مبدا برای ابرسطح با حداکثر مرز جداکننده است. اگر b صرفنظر شود، پاسخ تنها ابرسطح‌هایی هستند که از مبدا می‌گذرند. فاصله عمودی ابرسطح تا مبدا با تقسیم قدرمطلق مقدار پارامترb برطول w بدست می‌آید.

ایده اصلی این است که یک جداکننده مناسب انتخاب شود. منظور، جداکننده‌ای است کـه بیشـترین فاصله را با نقاط همسایه از هر دو طبقه دارد. این جواب درواقع بیشترین مرز را با نقاط مربوط بـه دو طبقه مختلف دارد و می‌تواند با دو ابرسطح موازی که حداقل از یکی از نقاط دو طبقه عبور می‌کننـد، کران‌دار شود. این بردارها، بردارهای پشتیبان نام دارند. فرمول ریاضی این دو ابرسطح موازی کـه مـرز جداکننده را تشکیل می‌دهند در عبارات زیر نشان داده شده است:

$$w.x - b = 1$$

$$w.x - b = -1$$

نکته قابل توجه این است که اگر داده‌های تعلیمی به‌صورت خطی تفکیـک‌پـذیر باشـند، مـی‌تـوان دو ابرسطح مرزی را به گونه‌ای انتخاب کرد که هیچ داده‌ای بین آنها نباشد و سپس فاصـله بـین ایـن دو ابرسطح موازی را به حداکثر رساند. با به کارگیری قضایای هندسی، فاصله این دو ابرسطح عبارت است از $2/|w|$ ، پس باید $|w|$ را به حداقل رساند. همچنین باید از قرار گرفتن نقاط داده در ناحیه درون مـرز جلوگیری کرد، برای این کار یک محدودیت ریاضی به تعریف فرمال اضافه مـی‌شـود. بـرای هـر i، بـا اعمال محدودیت‌های زیر اِطمینان حاصل می‌شود که هیچ نقطه‌ای در مرز قرار نمی‌گیرد:

برای داده‌های مربوط به طبقه اول $w.x_i - b \geq 1$

برای داده‌های مربوط به طبقه دوم $w.x_i - b \leq -1$

می‌توان این محدودیت را به‌صورت رابطه زیر نشان داد:

$$c_i(w.x_i - b) \geq 1 \qquad , 1 \leq i \leq n$$

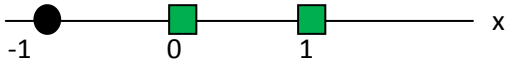
لذا مسأله بهینه‌سازی بدین شکل تعریف می‌شود:

به حداقل رساندن w، با درنظر گرفتن محدودیت زیر:

$$c_i(w.x_i - b) \geq 1 \qquad , 1 \leq i \leq n$$

مثال- حالا با یک مثال خیلی ساده نشان می‌دهیم که SVM چگونه ابرسطح را می‌سازد. فرض کنید ۳ نمونه داده داریم که می خواهیم آنها را تفکیک کنیم. داده ها یک بعدی هستند.

$$h = D(x)$$



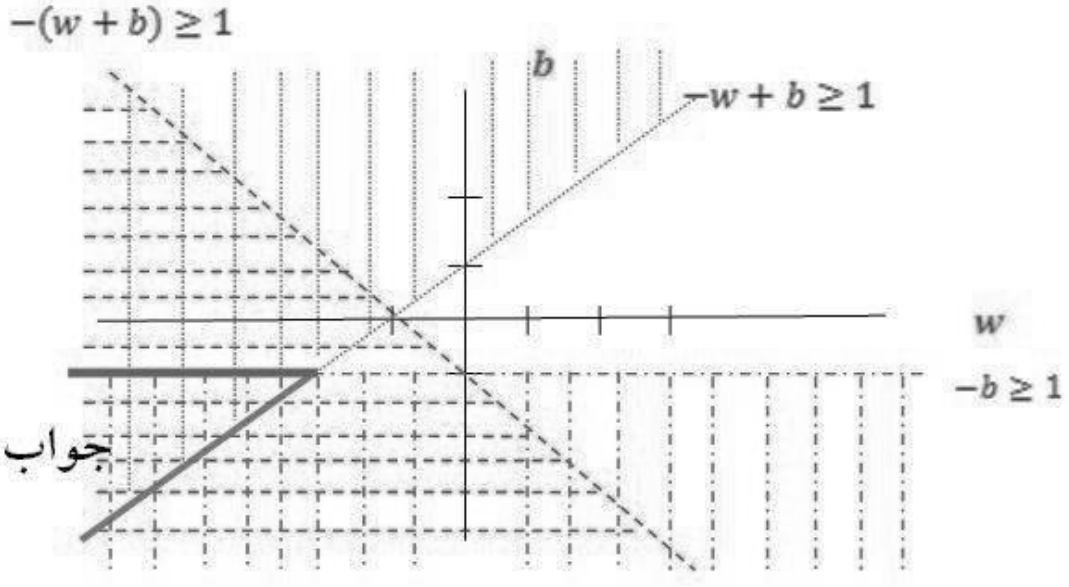
داده های ما چنانچه در بالا توضیح داده شد باید در رابطه زیر صدق کنند:

$$c_i(w.x_i - b) \geq 1 \qquad , 1 \leq i \leq n$$

$$-w + b \geq 1$$

$$-(w + b) \geq 1$$

$$-b \geq 1$$

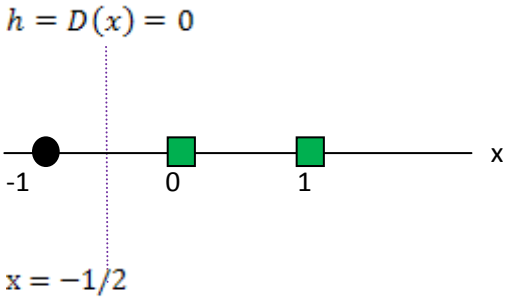


همانگونه که قبلاً هم گفتیم هدف مینیمم کردن $\|w\|^2$ است. با توجه به شکل بالا جواب به شکل زیر است:

$$w = -2 \quad b = -1$$

بنابراین تابع تصمیم(معادله ابرسطح) به شکل زیر خواهد بود:

$$D(x) = -2x - 1 = 0 \quad \rightarrow \quad x = -1/2$$



۳-۴- صورت اولیه مسأله

مسأله بهینه‌سازی نشان داده شده در قسمت قبل دشوار است چون وابسته به مقدار دقیق $|w|$ است. علت این است که از لحاظ ریاضی ، حل یک مسأله بهینه سازی غیرمحدب[1] بسیار دشوارتر از حل یک مسأله بهینه سازی محدب است. خوشبختانه می‌توان مسأله را با جایگزینی $\frac{1}{2}\|w\|^2$ به جای $|w|$ بدون تغییری در جواب، حل کرد. این مسأله از نوع مسائل برنامه‌نویسی درجه دوم[2] است. پس مسأله بهینه‌سازی به شکل زیر تبدیل می‌شود:

به حداقل رساندن $\frac{1}{2}\|w\|^2$ با درنظر گرفت محدودیت $c_i(w.\ x - b) \geq 1 \quad , 1 \leq i \leq n$

فاکتور $\frac{1}{2}$ جهت سادگی محاسبات ریاضی استفاده می‌شود. این مسأله را می‌توان با تکنیک‌های برنامه‌نویسی درجه دوم حل کرد.

۳-۵- نحوه حل مسأله در حالت کلی

مسأله، به حداقل رساندن تابع $f(x)$ مشروط به محدودیتِ $g(x) = 0$ است. شرط لازم برای اینکه x_0 یک جواب باشد به شرح زیر است:

$$\begin{cases} \frac{\partial}{\partial x}\left(f(x) + \alpha g(x)\right)|_{x=x_0} = 0 \\ g(x) = 0 \end{cases}$$

α ضریب لاگرانژ[3] است. برای چندین محدودیت $g_i(x) = 0$ ، $i=1,...,m$ ، نیاز به ضریب لاگرانژ α_i برای هر یک از محدودیت‌ها است. آنگاه رابطه بالا به شکل زیر نمایش داده می‌شود.

[1] Non-convex optimization problem
[2] quadratic programming (QP) optimization
[3] Lagrange multiplier

$$\begin{cases} \frac{\partial}{\partial x}(f(x) + \sum_{i=1}^{n} \alpha_i g_i(x))|_{x=x_0} = 0 \\ g_i(x) = 0 \qquad\qquad \text{for } i = 1, \dots, m \end{cases}$$

در حالتی که محدودیت به‌صورت یک نامساوی باشد، $g_i(x) \leq 0$ باز هم مسأله به همان صورت است تنها با این تفاوت که α_i باید مثبت باشد. در این حالت اگر x_0 جوابی برای مسأله بهینه‌سازی باشد، آنگاه باید برای هر $i = 1, \dots, m$ وجود داشته باشد $\alpha_i \geq 0$ به گونه‌ای که x_0 در شرایط زیر صدق کند.

$$\begin{cases} \frac{\partial}{\partial x}(f(x) + \sum_{i=1}^{n} \alpha_i g_i(x))|_{x=jx_0} = 0 \\ g_i(x) \leq 0 \qquad\qquad \text{for } i = 1, \dots, m \end{cases}$$

تابع $f(x) + \sum_{i=1}^{n} \alpha_i g_i(x)$ تابع لاگرانژی نام دارد. شیب این تابع باید مساوی صفر قرار داده شود.

۳-۶- حل مسأله اصلی

مسأله به حداقل رساندن $\frac{1}{2}\|w\|^2$ با در نظر گرفتن محدودیت $1 - y_i(w^T x_i + b) \leq 0$ برای $i = 1, \dots, n$ است. در این حالت تابع لاگرانژی عبارت است از:

$$l = \frac{1}{2} w^T w + \sum_{i=1}^{n} \alpha_i (1 - y_i(w^T x_i + b))$$

لازم به توجه است که $\mathbf{w^T w} = \mathbf{||w||^2}$ **است.** با مشتق‌گیری از l خواهیم داشت:

$$w + \sum_{i=1}^{n} \alpha_i (-y_i) x_i = 0 \rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \ , \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

۳-۷- مسأله همزاد[1]

اگر جایگزینی $w = \sum_{i=1}^{n} \alpha_i y_i x_i$ در l انجام شود، آنگاه رابطه زیر برقرار است:

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i x_i^T \sum_{j=1}^{n} \alpha_i y_i x_j + \sum_{i=1}^{n}\left(1 - y_i\left(\sum_{j=1}^{n} \alpha_i y_i x_j^T x_i + b\right)\right)$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n}(\alpha_i) - \sum_{i=1}^{n} \alpha_i y_i \sum_{j=1}^{n} \alpha_j y_j x_j^T x_i - b\sum_{j=1}^{n} \alpha_i y_i$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i$$

لازم به توجه است که $\sum_{i=1}^{n} \alpha_i y_i = 0$ برقرار است . لذا تابع هدف تنها یک تابع از α_i است. پس مسأله همزاد بدین شکل تعریف می‌شود:

$$\max.\ w(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1\ j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

با توجه به محدودیت $\alpha_i \geq 0$ (خاصیت α_i زمانیکه ضرایب لاگرانژ معرفی شدند) و شرط $\sum_{i=1}^{n} \alpha_i y_i = 0$ (نتیجه حاصل از مشتق گیری از تابع لاگرانژ). این مسأله، یک مسأله برنامه‌نویسی درجه دوم است و همواره یک مقدار بیشینه برای α_i وجود دارد.

در روش ماشین بردار پشتیبان، بردارهای ورودی به یک فضای چند بعدی نگاشت می‌شوند. پس از آن، یک ابرسطح ساخته خواهد شد که با حداکثر فاصله ممکن، بردارهای ورودی را از هم جدا خواهد

[1] Dual Problem

کرد. به این ابرسطح،" ابرسطح با حداکثر مرز جداکننده" گفته می‌شود. همانگونه که در شکل اول نشان داده شده است، دو ابرسطحِ موازی در دو سمتِ " ابرسطح با حداکثر مرز جداکننده" ساخته خواهد شد که داده‌های مربوط به دو طبقه را به گونه‌ای از هم مجزا می‌کنند که هیچ داده‌ای در مرز بین این دو ابرسطح قرار نمی‌گیرد. ،" ابرسطح با حداکثر مرز جداکننده"، ابرسطحی است که فاصله بین دو ابرسطحِ موازی را به حداکثر می‌رساند. فرض بر این است که هرچقدر مرز جداکننده یا در واقع، فاصله بین دو ابرسطح موازی بیشتر باشد، خطای طبقه‌بندی هم کمتر خواهد بود.

فصل چهارم:

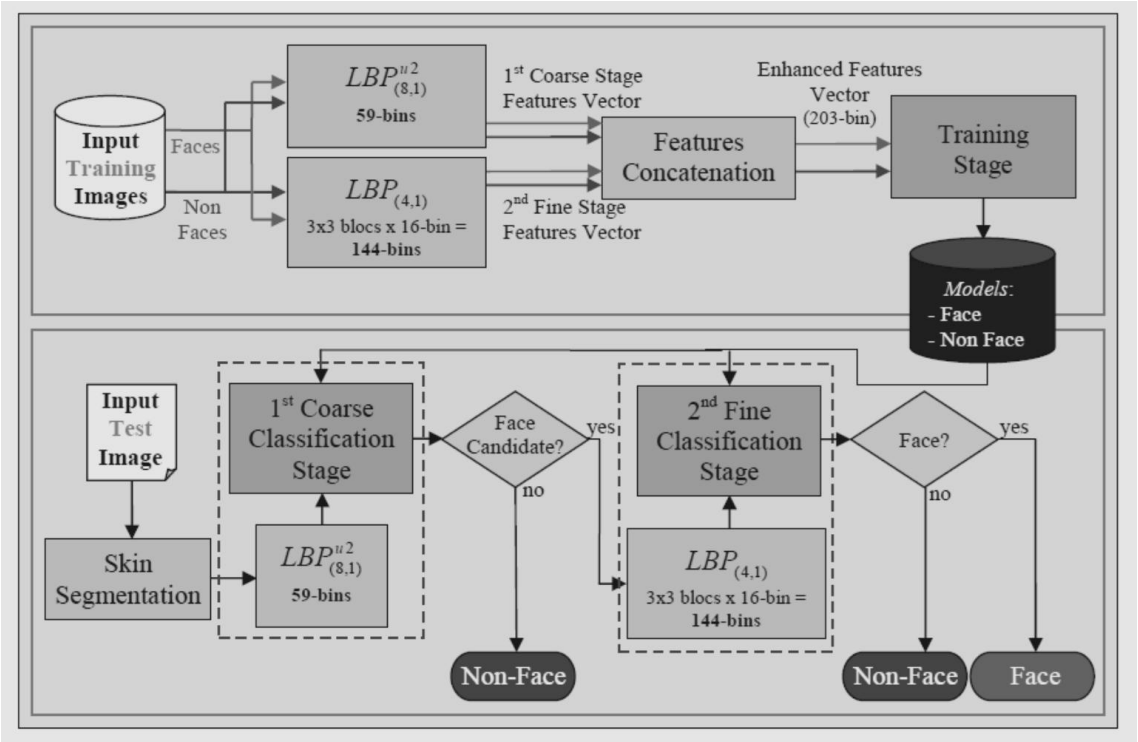
کشف چهره با استفاده از LBP و SVM

۴-۱- مقدمه

در این فصل به برنامه الگوی باینری محلی و همچنین نمایش صورت در سیستم تشخیص چهره را بررسی خواهیم کرد. اولین مرحله تفکیک درشت دانه‌ها برای انتخاب پیش‌نامزدهای صورت و دومین مرحله تفکیک ریز دانه‌ها برای مشخص کردن تصاویر غیر چهره.

شکل زیر معرفی کننده طرح کشف چهره عمومی برای این پروژه پژوهشی است، از جمله:

- مرحله آموزش (جعبه بالا): معرفی نمونه‌های آزمایشی چهره و غیر چهره در سیستم و محاسبه بردار ویژگی ریز و درشت (مراحل ۱ و ۲) به‌صورت موازی و جمع شدن در یک بردار منحصر به فرد پیشرفته ۲۰۳ ستونه برای توصیف هر چهره و غیر چهره و بعد از این تمام این نتایج برای تولید مدل متوسط مقدار برای هر کلاس مورد استفاده قرار خواهد گرفت.
- مرحله تست (کادر پایین): در این قسمت ابتدا مدل را با بردار ویژگی درشت مقایسه کرده و در صورت کشف چهره به مرحله بردار ویژگی ریز می رویم در غیر این‌صورت تصور مورد نظر چهره نبوده. تنها تصاویری چهره کشف داده خواهند شد که در هر دو بردار ویژگی چهره کشف داده شوند.



شکل ۴-۱ الگو کشف چهره

تصویر بالا نمای کلی از سیستم کشف چهره برای دادن دید کلی از این فرایند در نظر گرفته شده است، اگر چه برخی از فرایندها مانند اسکن تصویر ورودی و downsampling برای سادگی حذف شده‌اند. در هر مرحله قسمتی از بردار ویژگی LBP 203-bin (الگوی باینری محلی ۲۰۳ ستونه) استفاده می شود.

۴-۲- روش پویش بلوکی تصاویر

کاوش یا پویش تصویر برای جستجو چهره بر اساس روش پنجره کشویی و در مقیاس‌های مختلف انجام می‌شود. که این امکان را به ما می دهد تا چهره‌های صحیح‌تری در مکان خاصی با رزولیشن‌های مختلفی بدست آوریم.

پارامترهای زیر روش اسکن و همچنین تصمیم گیری در مورد موازنه بین سرعت و وضوح آشکارسازی چهره را شرح می‌دهد.

- اندازه بلوک: بلوک مربع یا مستطیلی شکل که تصویر را به‌صورت کشویی و در اندازه‌های مختلف پویش می‌کند.
- حرکت گام اسکن: تعداد پیکسل‌های که برای گام بعدی بلوک پویشگر تعریف می‌شود.
- نرخ Down-sampling: ضریب تغییر اندازه تصویر برای پویش‌های بعدی

توجه کنید که در هنگام جستجوی تصاویری با وضوح بالا گرفتن نرخ نمونه گیری پایین و گام اسکن حرکتی کوچک می تواند سبب کاهش قابل توجهی در الگوریتم کشف چهره ایجاد کند.

۴-۳- استخراج ویژگی برای دو مرحله طرح کشف صورت

با توجه به کار A.Hadid [5][6] درباره کشف چهره با استفاده از الگوی باینری محلی، پایه کشف چهره بر اساس دو مرحله اجرا می‌شود. دلیل اصلی برای انتخاب این طرح به منظور بهبود سرعت و بالا رفتن راندمان کشف بوده است.

همانطور که بعدا با جزئیات بیشتر توضیح داده خواهد شد، به جای استفاده از مقادیر پیکسل N*N (اندازه تصویر) برای توصیف یک تصویر، این نمایش جدید چهره برای دستیابی به نکات زیر ارائه شده است:

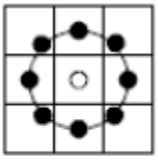
۱. تقلیل ابعاد: تنها ۲۰۳ مقدار مورد نیاز است و برای رزولیشن پایین‌تر تاثیر بیشتری دارد.
۲. کارایی در نمایش چهره نسبت به چالش‌های مختلف مانند تغییرات روشنایی و یا زاویه دید دارد.

جدول ۴-۱ کاهش طول توصیفگر چهره برای اندازه‌های استفاده شده در این تحقیق

Image size cases	Number of pixels	Reduction using 203 values
16x16	256	20,70 %
18x21	378	46,29 %
19x19	361	43,76 %

۴-۳-۱- استخراج ویژگی های مرحله اول

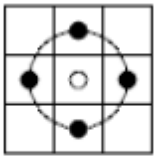
اگر تصویر بتواند کاندیدی برای چهره باشد در مرحله اول (درشت) تایید می شود. بعد از این مرحله $LBP^{u2}_{(P,R)}$ از کل تصویر یک برچسب هیستوگرام ۵۹ ستونه برای داشتن توصیف دقیقی از تصویر سراسری استخراج می‌کند.



شکل ۴-۲ الگوی باینری محلی (8.1)

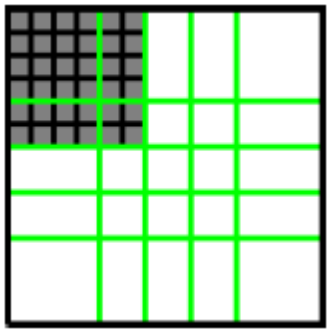
۴-۳-۲- استخراج ویژگی در مرحله دوم

تنها نتایج مثبت از مرحله اول در این مرحله مورد ارزیابی ریز قرار خواهد گرفت که کار بررسی پراکنش مکانی توصیف بافت را بر عهده دارد. در این حالت عملگر $LBP_{(4.1)}$ به کل تصویر ۱۶ پیکسل در ۱۶ پیکسل اعمال می‌شود و نتیجه بدست آمده یک تصویر ۱۴×۱۴ خواهد بود.



شکل ۴-۳ الگوی باینری محلی (4.1)

سپس تصویر به بلوک‌های ۳×۳ با اندازه ۶ پیکسل در ۶ پیکسل با تداخل ۲ پیکسل تقسیم می‌شود، همانطور که در شکل زیر نشان داده شده است بلوک خاکستری رنگ نشان دهنده اولین بلوک و خطوط پررنگ دیگر بلوک‌ها را نشان می دهند.



شکل ۴-۴ کشف چهره در مرحله دوم (بلوک‌های ۳×۳ با تداخل ۲ پیکسل)

با استفاده از تقسیم بندی بلوکی ۳×۳ هر بلوک با استفاده از برچسب هیستوگرام ۱۶ ستونه شرح مختصری از آن منطقه می دهد. در نتیجه یک مقدار از بردار ویژگی در این مرحله بدست می‌آید (۱۴۴=(۳×۳)×۱۶)، همانطور که بعداً دیده می‌شود وزن مختلف می‌تواند به هر منطقه در فاز طبقه‌بندی برای تاکید مناطق مهمتری از چهره اعمال شود. در مرحله آموزش نتیجه این دو برچسب هیستوگرام در یک بردار ویژگی پیشرفته بهم متصل هستند.

$$59 - bin + (3x3\ block\ x\ 16 - bin) = 59 - bin + 144 - bin = 203 - bin$$

در تصویر بعدی نتیجه استفاده از الگوی باینری محلی در مرحله ۱ و ۲ نشان داده شده است. در تصاویر زیر شما به وضوح مشاهده می گنید چرا به الگوی باینری محلی توصیفگر بافت می‌گویند.

شکل ۴-۵ تصویر اصلی

شکل ۴-۶ برچسب الگوی باینری محلی (8.1) – مرحله ۱

شکل ۴-۷ برچسب الگوی باینری محلی (4.1) – مرحله ۲

همانطور که در شکل بالا نشان داده شده، چارچوب خصوصیات چهره (چشم، دهان، بینی، ابرو، ...) به وضوح ملاحضه می‌شود. در مرحله اول خطوط برچسب تصویر بشدت برجسته هستند که دید کلی و مفیدی از تصاویر غیرچهره برای کشف در مرحله اول می‌دهند. از سویی دیگر در برچسب تصویر مرحله دوم اطلاعات بافت محلی مغیدتر و مفصل‌تر برای تصمیم گیری مرحله نهایی است که در آن تصویر غیر چهره به‌صورت محلی مورد بررسی قرار می گیرد.

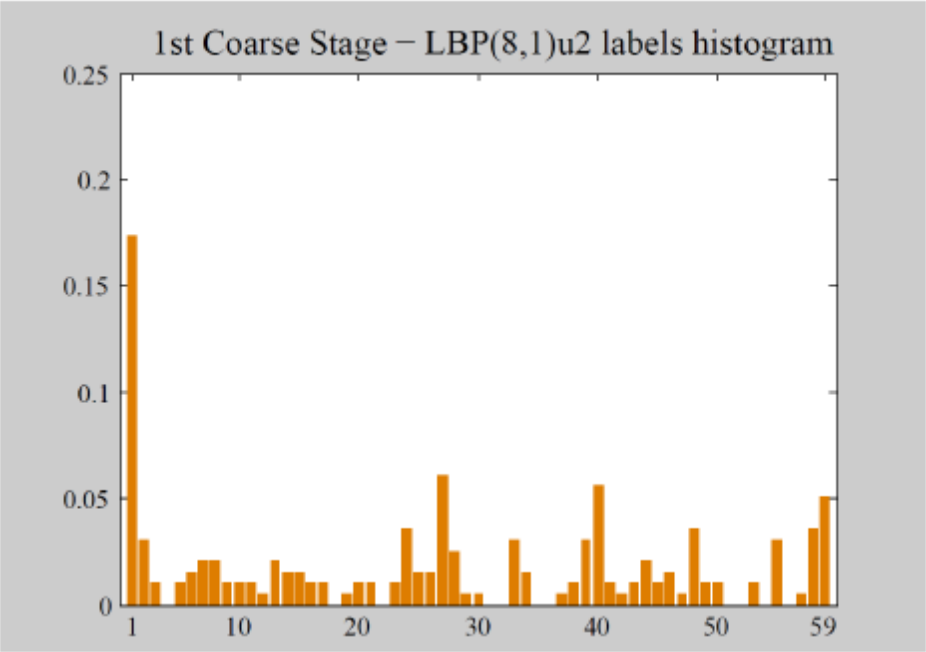
توجه کنید که در تصویر مرحله اول بافت کاملاً یکنواخت در سفید (۱۱۱۱۱۱۱۱) و سیاه (۰۰۰۰۰۰۰۰) نشان داده می‌شوند. آمار و ارقام زیر برای مقایسه برچسب هیستوگرام چهره و غیرچهره از هر دو مرحله در نظر گرفته شده‌اند. به ترتیب در تصاویر زیر اولین تصویر مربوط به نمونه

تصویر چهره ۱۶×۱۶ است، در تصویر بعد از آن تصویر نتیجه برچسب الگوی باینری محلی (8.1) می-باشد، در تصویر سوم هیستوگرام مربوط به آن با الگوی غیریکنواخت انباشته شده در ستون اول است و تصویر آخر هیستوگرام برچسب مرحله دوم است که ۱۶ بلوک ۳×۳ به‌صورت چسبان در هیستوگرام (4.1) نشان داده شده است.

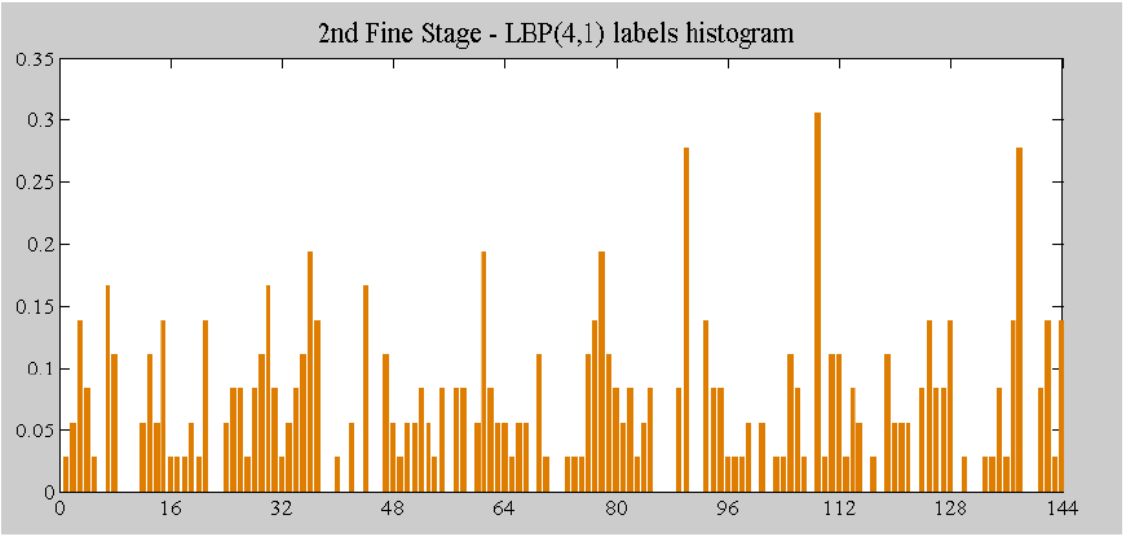
در شکل ... همین اطلاعات برای محاسبه یک تصویر غیر چهره نشان داده شده است.

شکل ۴-۸ نمونه چهره

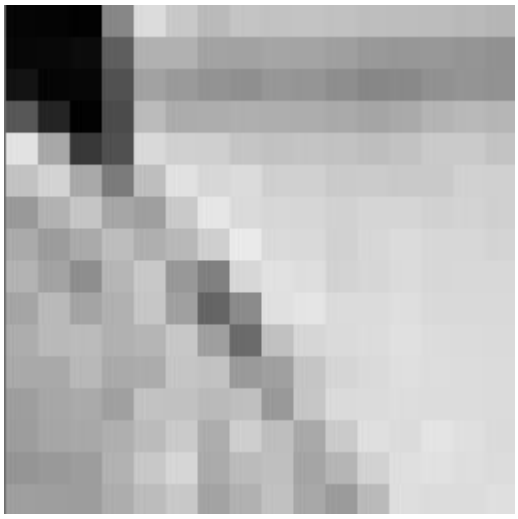
شکل ۴-۹ تصویر برچسب الگوی باینری محلی (8.1) مرحله اول



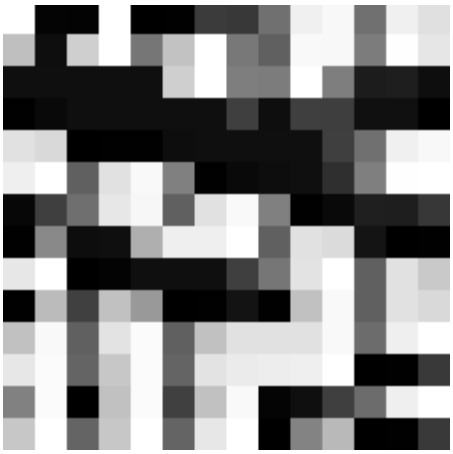
شکل ۴-۱۰ هیستوگرام برچسب الگوی باینری محلی (8.1) مرحله اول



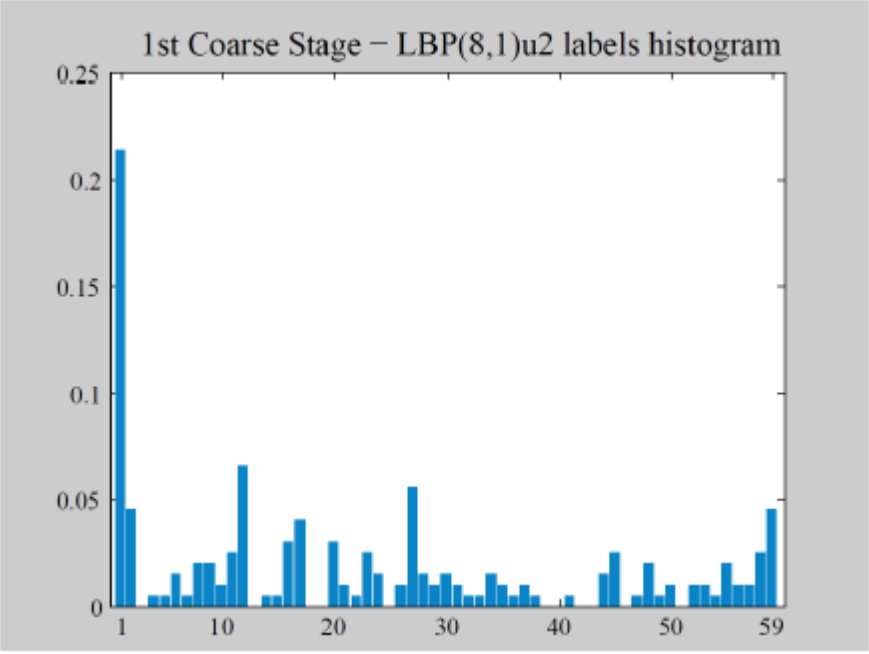
شکل ۴-۱۱ هیستوگرام برچسب الگوی باینری محلی (4.1) مرحله دوم



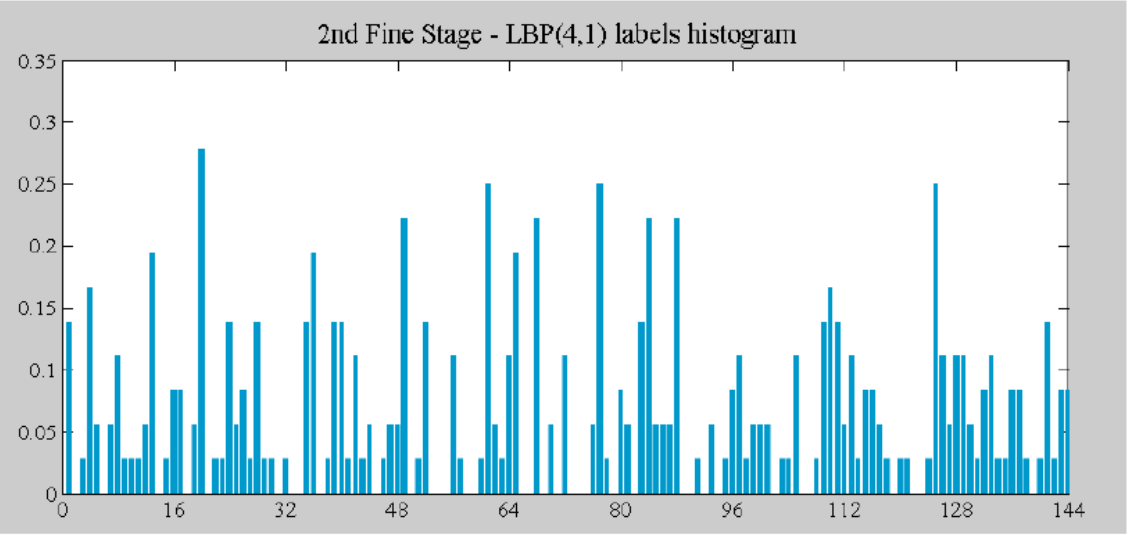
شکل ۴-۱۲ نمونه غیر چهره



شکل ۴-۱۳ تصویر برچسب الگوی باینری محلی (8.1) مرحله اول



شکل ۴-۱۴ هیستوگرام برچسب الگوی باینری محلی (8.1) مرحله اول



شکل ۴-۱۵ هیستوگرام برچسب الگوی باینری محلی (4.1) مرحله دوم

مقایسه هر دو نمونه می تواند این را تایید کند که اکثر برچسب‌های مهم در حالت الگوی باینری محلی (8.1) مربوط به الگوهای یکنواخت هستند، همچنین این را می توان در هیستوگرام برچسب الگوی باینری محلی (8.1) مرحله اول نیز مشاهده کرد، جایی که ستون اول شامل درصدی از الگوهای غیر یکنواخت اند. در نگاه اول تمیز کردن تفاوت‌های دو هیستوگرام بسیار مشکل است اما با این حال برای فرق گذاشتن در کشف چهره و غیر چهره در بردار ویژگی مفید هستند. برای جدا کردن این هیستوگرام‌ها از هم از روشی استفاده شده که در بخش بعدی آمده است.

۴-۴- کاربرد SVM در طبقه بندی کشف صورت

هدف از انجام این کار اجازه به کار گیری در موارد خاص کشف چهره را به ما می‌دهد. بر اساس آزمایشات A.hadid [6] یک تابع چند جمله‌ای درجه دوم به عنوان نوع سطح تصمیم‌گیری انتخاب شده است. الگوی باینری محلی در هر تکرار، LBP_x از زیر پنجره و تغذیه برای تعیین چهره و غیرچهره توسط SVM را محاسبه می‌کند. طبقه‌بند کننده در مورد چهره بودن یا نبودن در زیرپنجره با توجه به علامت تابع زیر تصمیم گیری می‌کند.

$$f(LBP) = sing(\sum_{i}^{l} a_i y_i K(LBP_x, LBP_{ti}) + b)$$

در جایی که:

- LBP_{ti} نمایش LBP از نمونه آزمایشی ti است.
- yi بسته به اینکه ti مثبت یا منفی است (چهره است یا نه) ۱ و ۱- خواهد بود.
- l شماره نمونه آزمایش ماست.
- b یک کمیت (پایه) است.
- K دومین تابع چندجمله‌ای درجه دوم است.

$$K(LBP_x, LBP_{ti}) = (1 + LBP_x . LBP_{ti})^2$$

- αi پارامترهای مشخص طبقه‌بند SVM می باشد که با برنامه نویسی درجه دوم زیر بهبود میابد.

$$Max(\sum_{i=1}^{l} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(LBP_{ti}, LBP_{tj}))$$

$$Subject \quad to: \left\{ \sum_{i=1}^{l} \alpha_i y_i = 0, \quad 0 \le \alpha_i \le C \right.$$

۴-۵- مرور کلی بر ماشین‌های بردار پشتیبان SVM

داده‌های آموزشی $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ را در نظر می‌گیریم که در آن، ورودی $\mathbf{x}_i \in \Re^m$ یک بردار ویژگی و خروجی $y_i \in \{\pm 1\}$ برچسب کلاس آن است. در روش SVM، ابتدا توسط نگاشت غیرخطی $\Phi : \Re^m \to F$ ورودی $\mathbf{x}$ به $\mathbf{z} = \Phi(\mathbf{x})$ در فضای هیلبرت[1] F با ابعاد بالاتر نگاشته می‌شود (با ضرب داخلی $\langle \cdot, \cdot \rangle$). حالتی را در نظر می‌گیریم که دو کلاس به‌صورت خطی در F تفکیک‌پذیرند، یعنی بردار $\mathbf{w} \in F$ و اسکالر $b \in \Re$ وجود دارند به‌طوریکه برای تمام اعضا D: $y_i(\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle + b) \ge 1$.

[1] Hilbert space

از میان تمام ابرصفحه‌های $(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b)$، ابرصفحه‌ای بهینه است که دارای بیشترین فاصله با نزدیک‌ترین داده‌های مثبت و منفی می‌باشد. نشان داده شده است که $\mathbf{w}$ این ابرصفحه‌ی بهینه را می‌توان با مینیمم‌سازی $\|\mathbf{w}\|$ بدست آورد که به رابطه‌ی $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)\,;\, \alpha_i \geq 0$ منجر می‌شود. بردار α_iها، $\mathbf{\Lambda} = (\alpha_1, \ldots, \alpha_N)^{\mathrm{T}}$، را می‌توان با حل مسئله‌ی برنامه‌نویسی درجه‌دوم[1] زیر یافت:

$$\max_{\mathbf{\Lambda}} W(\mathbf{\Lambda}) = \mathbf{\Lambda}^{\mathrm{T}}\mathbf{1} - \frac{1}{2}\mathbf{\Lambda}^{\mathrm{T}}\mathbf{Q}\mathbf{\Lambda} \quad \text{subject to} \quad \mathbf{\Lambda} \geq \mathbf{0} \;;\; \mathbf{\Lambda}^{\mathrm{T}}\mathbf{Y} = \mathbf{0}$$

که در آن $\mathbf{Y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ و $\mathbf{Q}$ یک ماتریس متقارن با درایه‌هایی به‌صورت زیر است:

$$Q_{i,j} = y_i y_j \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

ابعاد فضای F و در نتیجه $\Phi(\mathbf{x}_i)$ و $\Phi(\mathbf{x}_j)$ در (ب-۲) معمولا بسیار بالاست. ویژگی بارز SVM که این روش را عملی می‌کند آن است که، بدون نیاز به محاسبه‌ی صریح $\Phi(\mathbf{x}_i)$ و $\Phi(\mathbf{x}_j)$ می‌توان $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$ را در (ب-۲) بدست آورد. با در نظر گرفتن تابع کرنل به‌صورت:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

می‌توان نشان داد که از هر تابعی که تئوری مرسر[2] را برآورده کند، می‌توان به عنوان کرنل استفاده کرد و برای آن کرنل یک نگاشت مرتبط Φ وجود دارد به‌طوریکه رابطه‌ی (ب-۳) برقرار است. از جمله توابعی که شرایط مرسر را برآورده کرده و معمولا استفاده می‌شوند می‌توان به کرنل خطی $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$، کرنل گوسی $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \| \mathbf{x}_i - \mathbf{x}_j \|^2)$ و کرنل چندجمله‌ای $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + 1)^d$ اشاره کرد.

در مرحله آزمایش، تابع تصمیم‌گیری برای داده‌ی $\mathbf{x} \in \Re^m$ به‌صورت زیر خواهد بود:

[1] Quadratic programming
[2] Mercer's theorem

$$f(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b$$

به این ترتیب کلاس داده $\mathbf{x}$ توسط تابع $\text{sign}(f(\mathbf{x}) - f_0)$ مشخص می‌گردد که آستانه f_0 معمولا صفر در نظر گرفته می‌شود.

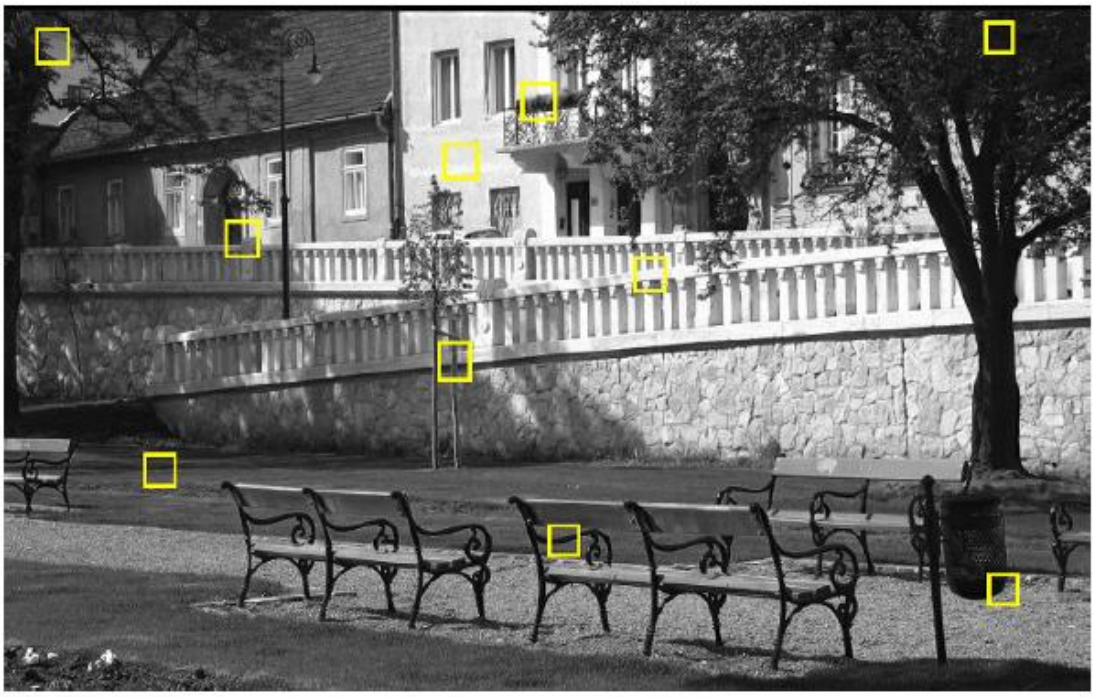
زمانی که داده‌های آموزشی در فضای F تفکیک‌پذیر نباشند، سعی بر مینیمم‌سازی $\|\mathbf{w}\|$ و در عین حال تفکیک داده‌ها با کمترین تعداد خطا می‌شود. در این حالت قید اول رابطه‌ی (ب-۱) به‌صورت $\mathbf{0} \le \mathbf{\Lambda} \le C\mathbf{1}$ در می‌آید که در آن C توازن بین ماکزیمم‌سازی حاشیه و مینیمم‌سازی خطای آموزشی را کنترل می‌کند.

SVM اساسا یک طبقه‌بند باینری است. دو روش معمول برای استفاده از SVM در مسائل M-کلاسه وجود دارد. در روش اول که "یکی در برابر همه" نام دارد، یک SVM برای هرکلاس با متمایز کردن آن نسبت به $(M-1)$ کلاس دیگر ساخته می‌شود. بنابراین تعداد SVM ها برابر با M خواهد بود. در نهایت کلاسی که SVM متناظر آن بیشترین مقدار تابع تصمیم‌گیری (ب-۴) را دارد انتخاب می‌شود. در روش دوم که "یکی در برابر یکی" نام دارد، یک SVM برای هر جفت از کلاس‌ها ساخته می‌شود. بنابراین تعداد SVM ها برابر با $M(M-1)/2$ خواهد بود. به ازای هر SVM یک رای به کلاس برنده اضافه می‌گردد و نهایتا کلاس با بیشترین رای انتخاب می‌گردد.

۴-۶- مرحله آموزش

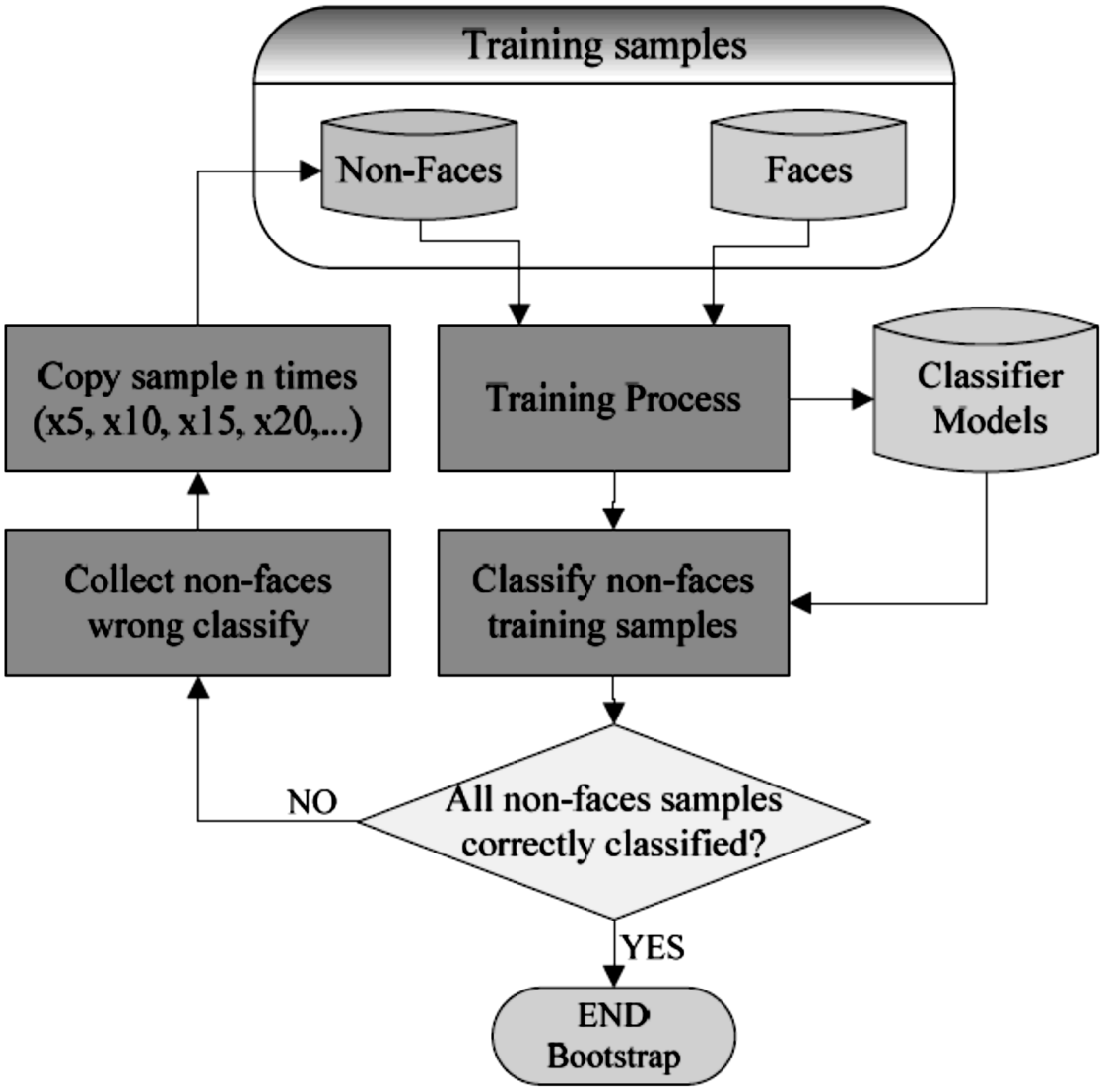
همانطور که در مرحله آموزش چهره توسط A.Hadid [6] و K.-K. Sung [14] پیشنهاد شده در قسمت آموزش روش کشف چهره نمونه‌های منفی توسط روشی به نام بوت استرپ جمع آوری می‌شوند که به‌صورت وارد کردن کشف‌های غلط در مجموعه‌های آموزشی به‌صورت پیاپی است. با استفاده از روش مذکور از مشکل انتخاب دستی نمونه‌های آموزشی منفی مناسب پرهیز می‌شود.

در آخر هر مرحله‌ی بوت استراپ مجموعه نمونه‌های آموزشی منفی با وارد شدن تصاویر غیر چهره جدیدی که در مرحله قبل غلط طبقه بندی شده بودند بزرگتر می‌شود. در آزمایشات ما به منظور جمع آوری الگوهای غیر چهره از چنین استراتژی استفاده می‌شود. بنابراین در مرحله اول ما بین ۶۰۰۰ تا ۷۰۰۰ الگوی غیر چهره به‌صورت کاملاً تصادفی از مجموعه تصاویر طبیعی موجود در اینترنت که شامل چهره نمی‌شوند استخراج شده است.



شکل ۴-۱۶ تصاویر طبیعی که از آن نمونه آموزشی منفی استخراج شده است.

سپس طرح زیر را انقدر انجام می دهیم تا خطاهای مربوط به کشف غلط چهره مانند تصویر بالا به حداقل برسد.



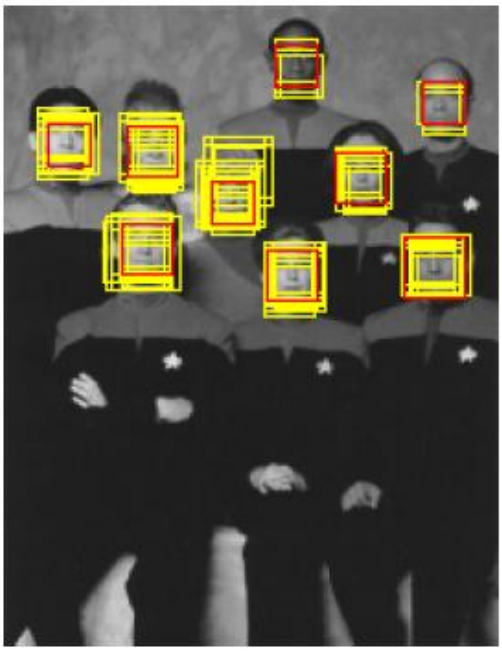
شکل ۴-۱۷ فلوچارت استراتژی بوت استراپ

۴-۷- حذف کشف‌های کنار هم

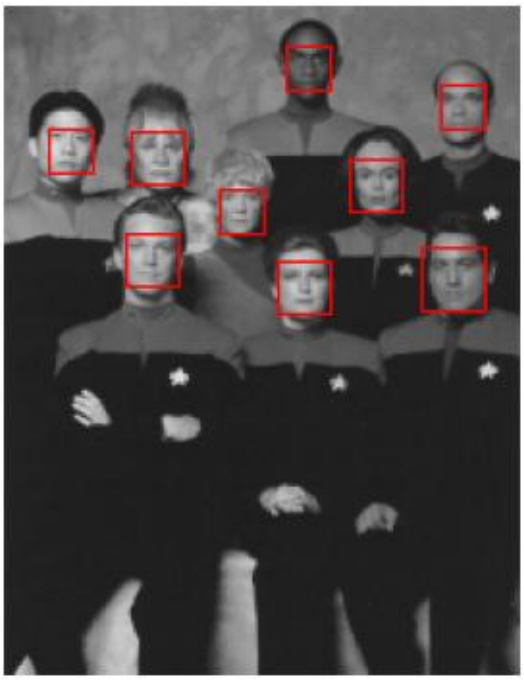
در روند کشف چهره ارائه شده، تصویر ورودی توسط یک پنجره مستطیلی یا مربعی اسکن می‌شود که تعیین کننده وضوح روش کشف چهره اند. همچنین گامی برای این پنجره متحرک و یک نرخ down-sampling برای جستجو تصویر در اندازه‌های مختلف در نظر گرفته شده است.

در نتیجه روش ارائه شده اغلب چندین چهره بگونه ای در مکان‌ها و اندازه‌های مختلف کنار هم قرار دارند کشف می‌شوند که باعث پایین آمدن میزان اطمینان پذیری روش ارائه شده شود. در مقالات رایج‌ترین روش‌های حل مشکلات کشف‌های هم پوشان در اطراف یک چهره دو مورد زیر است:

- حذف کشف‌های تداخل در هم: کشف‌هایی که در یک مکان با تراکم کم جمع شده اند به احتمال زیاد کشف‌ها اشتباه بوده، بنابراین می‌توانند حذف شوند.
- ادغام کشف‌های تداخل در هم:
  - تمام کشف‌ها در یک مربع که مرکزیت کلیه کشف‌ها می‌باشد در نظر گرفته می‌شود.
  - تمام کشف‌ها در یک مربع بزرگ که تمام کشف‌ها را در خود جای می‌دهد قرار می‌گیرند.



شکل ۴-۱۸رنگ زرد کلیه کشف‌ها و رنگ قرمز انتخابی SVM.



شکل ۴-۱۹ حذف نتایجی که با هم تداخل دارند.

فصل پنجم:

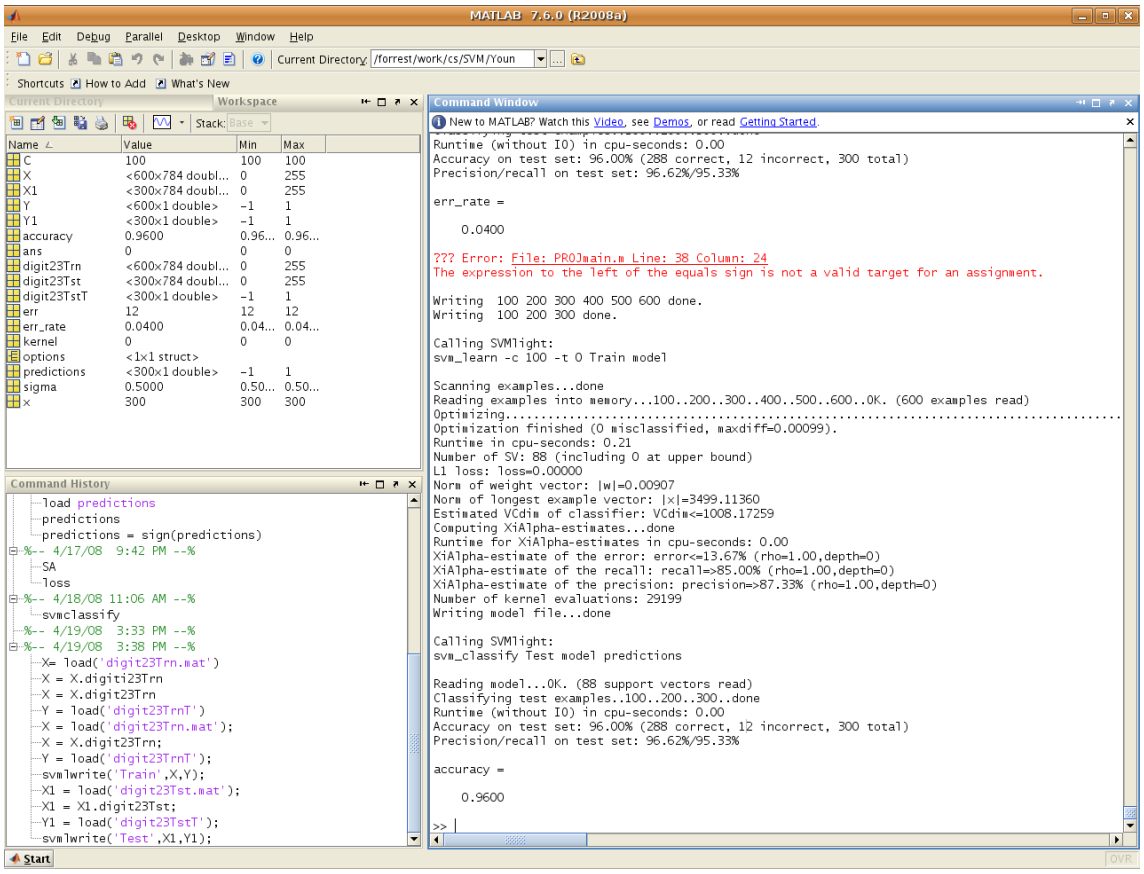
شبیه سازی و نتایج

۵-۱- مقدمه

شبیه‌سازی به وسیله نرم‌افزار شبیه‌ساز متلب انجام شده، در این فصل ابتدا به معرفی نرم‌افزار متلب پرداخته می‌شود سپس در مورد شبیه‌سازی و زیر برنامه‌های آن توضیح داده خواهد شد و در پایان نیز نتایج شبیه‌سازی.

۵-۲- نرم‌افزار متلب

متلب یک محیط نرم‌افزاری برای انجام محاسبات عددی و یک زبان برنامه‌نویسی نسل چهارم است. واژه متلب هم به معنی محیط محاسبات رقمی و هم به معنی خود زبان برنامه‌نویسی مربوطه است که از ترکیب دو واژه ماتریس و آزمایشگاه ایجاد شده‌است. این نام حاکی از رویکرد ماتریس محور برنامه‌است، که در آن حتی اعداد منفرد هم به عنوان ماتریس در نظر گرفته می‌شوند.



شکل ۵-۱ محیط متلب

کار کردن با ماتریس‌ها د ر متلب بسیار ساده است. در حقیقت تمام داده‌ها در متلب به شکل یک ماتریس ذخیره می‌شوند. برای مثال یک عدد (اسکالر) به شکل یک ماتریس ۱×۱ ذخیره می‌شود. یک رشته مانند «Whale is the biggest animal» به شکل ماتریسی با یک سطر و چندین ستون (که تعداد ستون‌ها به تعداد کاراکترهاست) ذخیره می‌شود. حتی یک تصویر به شکل یک ماتریس سه بعدی ذخیره می‌گردد که بُعد اول و دوم آن برای تعیین مختصات نقاط و بُعد سوم آن برای تعیین رنگ نقاط استفاده می‌شود. فایل‌های صوتی نیز در متلب به شکل ماتریس‌های تک ستون (بردارهای ستونی) ذخیره می‌شوند؛ بنابراین جای تعجب نیست که متلب مخفف عبارت آزمایشگاه ماتریس باشد.

علاوه بر توابع فراوانی که خود متلب دارد، برنامه‌نویس نیز می‌تواند توابع جدید تعریف کند.

ساخت رابط گرافیکی کاربر مانند دیالوگ‌هایی که در محیط‌های ویژوال مانند بیسیک و C وجود دارند، در متلب امکان‌پذیر است. این قابلیت، ارتباط بهتری را میان برنامه‌های کاربردی نوشته‌شده با متلب و کاربران برقرار می‌کند.

متلب که از محصولات شرکت مت‌ورکس است، برای گروه‌های مختلف مهندسان رشته‌های مختلف از جمله مهندسی برق، مکانیک، رایانهو... کاربرد بسیاری دارد. شرکت سازنده متلب شرکت مت‌ورکس نام دارد. این شرکت در سال ۱۹۸۴ بنیان نهاده شد و هم اکنون دارای بیش از ۲۰۰۰ نفر پرسنل است. دفتر مرکزی این شرکت در شهر Natick در ایالت ماساچوست آمریکا قرار دارد.

هسته متلب برای سرعت و کارایی بالا به زبان سی نوشته شده‌است ولی رابط گرافیکی آن به زبان جاوا پیاده‌سازی گشته‌است. برنامه‌های متلب اکثراً متن باز هستند و در واقع متلب (مانند بیسیک) مفسر (رایانه) است نه کامپایلر .قدرت متلب از انعطاف‌پذیری آن و راحت بودن کار با آن ناشی می‌شود، همچنین شرکت سازنده و گروه‌های مختلف، از جمله دانشگاه‌های سرتاسر جهان و برخی شرکت‌های مهندسی هر ساله جعبه‌ابزارهایخاص-کاربردی به آن می‌افزایند که باعث افزایش کارآیی و محبوبیت آن شده‌است. فهرستی از این جعبه‌ابزارها در زیر آمده‌است:

- سیمیولینک، ابزاری برای شبیه‌سازی سامانه‌ها به‌صورت مجرد
- جعبه‌ابزار مخابرات متلب، توابع و ابزارهای محاسبات مهندسی مخابرات
- جعبه‌ابزار کنترل متلب، توابع و ابزارهای محاسبات مهندسی کنترل
- جعبه‌ابزار فازی متلب، توابع و ابزارهای محاسبات فازی
- جعبه‌ابزار محاسبات متلب، توابع و ابزارهای محاسبات عددی
- جعبه‌ابزار تخمین متلب، توابع و ابزارهای محاسبات بحث تخمین سیستم در مهندسی کنترل
- جعبه‌ابزار آمار متلب، توابع و ابزارهای محاسبات آمار
- جعبه‌ابزار جمع‌آوری داده متلب، توابع و ابزارهای جمع‌آوری داده

- جعبه‌ابزار شبکه عصبی متلب، توابع و ابزارهای محاسبات شبکه عصبی
- جعبه‌ابزار پردازش تصویر متلب، توابع و ابزارهای محاسبات پردازش تصویر
- جعبه‌ابزار پردازش صوت متلب، توابع و ابزارهای محاسبات پردازش صوت
- جعبه‌ابزار احتمالات متلب
- جعبه‌ابزار محاسبات سیمبولیک متلب
- جعبه‌ابزار کارگاه بی‌درنگ متلب، توابع و ابزارهای محاسبات سامانه‌های بی‌درنگ

۵-۳- کدهای برنامه

کدهای برنامه در ۵ قسمت نوشته شده‌اند، قسمت main، lbp، image_load و model load و getmapping.

۵-۳-۱- الگوی باینری محلی

```
function LBP_map = lbp(I, R, P, phi)
spoints = zeros(P,2);
% Angle step.
a = 2*pi/P;
for i = 1:P
   spoints(i,1) = -R*sin((i-1)*a + phi);
   spoints(i,2) =  R*cos((i-1)*a + phi);
end
%   spoints = [0 1; -1 1; -1 0; -1 -1; 0 -1; 1 -1; 1 0; 1 1];
% Determine the dimensions of the input image.
[ysize xsize] = size(I);

miny=min(spoints(:,1));
maxy=max(spoints(:,1));
minx=min(spoints(:,2));
maxx=max(spoints(:,2));
% Block size, each LBP code is computed within a block of size bsizey*bsizex
bsizey=ceil(max(maxy,0))-floor(min(miny,0))+1;
bsizex=ceil(max(maxx,0))-floor(min(minx,0))+1;
% Coordinates of origin (0,0) in the block
origy=1-floor(min(miny,0));
origx=1-floor(min(minx,0));
% Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
```

```
% Fill the center pixel matrix C.
C = I(origy:origy+dy,origx:origx+dx);
% Initialize the result matrix with zeros.
siz = size(C);
D = zeros([siz,P]);
%Compute the LBP code image
for i = 1:P
   y = spoints(i,1)+origy;
   x = spoints(i,2)+origx;
   % Calculate floors, ceils and rounds for the x and y.
   fy = floor(y); cy = ceil(y); ry = round(y);
   fx = floor(x); cx = ceil(x); rx = round(x);
   % Check if interpolation is needed.
   if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
      % Interpolation is not needed, use original datatypes
      D(:,:,i) = I(ry:ry+dy,rx:rx+dx) - C;
   else
      % Interpolation needed, use double type images
      ty = y - fy;
      tx = x - fx;
      % Calculate the interpolation weights.
      w1 = (1 - tx) * (1 - ty);
      w2 =      tx  * (1 - ty);
      w3 = (1 - tx) *      ty ;
      w4 =      tx  *      ty ;
      % Compute interpolated pixel values
      N = w1*I(fy:fy+dy,fx:fx+dx) + w2*I(fy:fy+dy,cx:cx+dx) + ...
         w3*I(cy:cy+dy,fx:fx+dx) + w4*I(cy:cy+dy,cx:cx+dx);
      D(:,:,i) = N - C;
   end
end
sign_D = ( D>=0 );
%
=============================================================
```

```
LBP_map = zeros(siz);
for i = 1:P
   v = 2^(i-1);
   LBP_map = LBP_map + v*sign_D(:,:,i);
end
end
```

۵-۳-۲- بارگذاری تصاویر

```
function image = image_load
flag = 0;
while flag == 0
   [FileName , PathName] = uigetfile( ...
      {'*.gif; *.bmp; *.jpg; *.png; *.tif', 'Picture Files (*.gif,*.bmp,*.jpg,*.png,*.tif,...)';
      '*.gif', 'Graphics Interchange file (*.gif)';
      '*.bmp', 'Bitmap file (*.bmp)';
      '*.jpg; *.jpeg', 'JPEG image (*.jpg,*.jpeg)';
      '*.png', 'Portable Network Graphics file (*.png)';
      '*.tif; *.tiff', 'Tagged Image File Format (*.tif,*.tiff)';
      '*.*', 'All Files (*.*)'}, ...
      'Load Image');
   flag = 1;
   if FileName ~= 0
      oldFolder1 = cd('C:\Program
Files\MATLAB\R2012a\toolbox\matlab\imagesci\private');
      [format, ~] = imftype([PathName,FileName]);
      cd(oldFolder1)
      if (isempty(format))
         h = errordlg('Please Select A Valid Image File (*.jpg, *.gif, ...) !?', 'Invalid
Image File', 'replace');
         flag = 0;
      else
         oldFolder2 = cd(PathName);
         [X, map] = imread(FileName);
```

```
        info = imfinfo(FileName);
        if strcmp(info.ColorType ,'grayscale')
           image = X;
        elseif strcmp(info.ColorType ,'truecolor')
           image = rgb2gray(X);
        else strcmp(info.ColorType ,'indexed')
           image = ind2gray(X, map);
        end
        cd(oldFolder2)
     end
  else
     image = [];
  end
end
if exist('h','var')
  close(h)
end
end
```

۵-۳-۳- بار کردن مدل

```
function [Model, Max, Min, A, miu] = model_load
flag = 0;
while flag == 0
    [FileName , PathName] = uigetfile( ...
        {'*.mat', 'Mat-Files( *.mat)';
        '*.*', 'All Files (*.*)'}, ...
        'Load Model');
    flag = 1;
    if FileName ~= 0
        oldFolder = cd(PathName);
        [~, ~, ext] = fileparts(FileName);
        if strcmp(ext ,'.mat')
            temp = load(FileName);
```

```
            s = fieldnames(temp);
            if ~(all( ismember({'Model','Max','Min'},s)) )
                h1 = errordlg('Please Select A Valid *.mat
File !?', 'Invalid .mat File', 'replace');
                flag = 0;
            else
                Model = temp.Model;
                Max = temp.Max;
                Min = temp.Min;
                A = temp.A;
                miu = temp.miu;
            end
        else
            h2 = errordlg('Please Select A Valid Model File
(*.mat) !?', 'Invalid Model File', 'replace');
            flag = 0;
        end
        cd(oldFolder)
    else
        Model = [];
        Max = [];
        Min = [];
        A = [];
        miu = [];
    end
end
if exist('h1','var')
    close(h1)
elseif exist('h2','var')
    close(h2)
end
end
```

۵-۳-۴- تابع پویشگر تصویر

```
%  structure containing a mapping table for
%  LBP codes in a neighbourhood of SAMPLES sampling
%  points. Possible values for MAPPINGTYPE are
%       'u2'   for uniform LBP
%       'ri'   for rotation-invariant LBP
%       'riu2' for uniform rotation-invariant LBP.

function mapping = getmapping(samples,mappingtype)
table = 0:2^samples-1;
newMax  = 0;                                % number of
patterns in the resulting LBP code
index   = 0;
% ----------------------Uniform 2----------------------
if strcmp(mappingtype,'u2')
  newMax = samples*(samples-1) + 3;        % =
samples*(samples-1) + 2(full_zero & full-one) + 1(all non-
uniform)
  for i = 0:2^samples-1
    j = bitset(bitshift(i,1,samples),1,bitget(i,samples));
%rotate left
% number of 1->0 and 0->1 transitions
    numt = sum(bitget(bitxor(i,j),1:samples));
% in binary string x is equal to

% number of 1-bits in XOR(x,Rotateleft(x))
    if numt <= 2
      table(i+1) = index;
      index = index + 1;
    else
      table(i+1) = newMax - 1;
    end
```

```
  end
end
% --------------------Rotation invariant------------------
if strcmp(mappingtype,'ri')
  tmpMap = zeros(2^samples,1) - 1;
  for i = 0:2^samples-1
    r_min = i;
    r  = i;
    for j = 1:samples-1
      r =
bitset(bitshift(r,1,samples),1,bitget(r,samples)); %rotate
left
      if r < r_min
         r_min = r;
      end
    end
    if tmpMap(r_min+1) < 0
       tmpMap(r_min+1) = newMax;
       newMax = newMax + 1;
    end
    table(i+1) = tmpMap(r_min+1);
  end
end
% ---------------------------Uniform & Rotation invariant-
----------------
if strcmp(mappingtype,'riu2')
  newMax = samples + 3;
  for i = 0:2^samples - 1
    j = bitset(bitshift(i,1,samples),1,bitget(i,samples));
%rotate left
    numt = sum(bitget(bitxor(i,j),1:samples));
    if numt <= 2
      table(i+1) = sum(bitget(i,1:samples));
    elseif numt == 4
```

```
        table(i+1) = samples+1;
    else
      table(i+1) = samples+2;
    end
  end
end
%-------------------------------------------------------------
---------------
mapping.table = table;
mapping.samples = samples;
mapping.num = newMax;
```

۵-۳-۵- کد اصلی شبیه‌ساز

```
clear;
clc;
% ========================================================
I = double(image_load);
% --------------------------------------------------------------------------------------------
[siz1 siz2] = size(I);
image_stack{1} = I;
% --------------------------------------------------------------------------------------------
dialog1_text = {'Please input the Scaling Ratio (1~2):'};
dialog1_title = 'Input Scaling Ratio';
default_factor = {'1.2'};
options.Resize = 'on';
options.WindowStyle = 'normal';
answer = inputdlg(dialog1_text, dialog1_title, 1, default_factor, options);
scaling_ratio = str2num(answer{1});
% --------------------------------------------------------------------------------------------
mapping = getmapping(8, 'u2');
% Distance_measure = @ChiSquare;   % @Bhattacharyya
Final_DecValue = zeros([siz1 siz2]);
```

```
Max_DecValue = zeros([siz1 siz2]);
Final_Label = zeros([siz1 siz2]);
Final_Scale = zeros([siz1 siz2]);
% ---------------------------------------------------------------------------------------------
window = 19;
w = (window-1)/2;   ww = w - 1;
step = 2;
N_block = 9;
Blocks = cell(1, N_block);
% ---------------------------------------------------------------------------------------------
scale_power = 1;
while min(size(image_stack{scale_power})) / scaling_ratio > 19
   image_stack{scale_power+1} = imresize(image_stack{scale_power},
1/scaling_ratio, 'bilinear');
   scale_power = scale_power + 1;
end
% ---------------------------------------------------------------------------------------------
decision_matrix = cell(1,numel(image_stack));
Label = cell(1,numel(image_stack));
[Model, Max, Min, A, miu] = model_load;
OO = [];
tic;
for N_scale = 1:numel(image_stack)
   image = round(image_stack{N_scale});
   [M N] = size(image);
   % ---------------------------------------------------------------------------------------------
   figure; imshow(image, []);
   title(['Scale ',num2str(N_scale)], 'FontSize', 12)
   cah = gca;
   hold(cah,'on')
   % ---------------------------------------
   Box1 = lbp(image, 1, 4, 0);
   Box2 = lbp(image, 1, 8, 0);
   Box2  = mapping.table (Box2+1);
```

```
    %
=============================================================
    index_row = ww+1 : step : (M-2)-ww;
    index_col = ww+1 : step : (N-2)-ww;
    testNum = numel(index_col); % numel(index_row)*numel(index_col);
    % trainNum = size(trains, 1);
    % ---------------------------------------------------------------------------------------------
    ClassIDs = double( (rand(testNum, 1) > 0.5) );     % Unknown
    % ---------------------------------------------------------------------------------------------
    decision_matrix{N_scale} = zeros(M, N);
    Label{N_scale} = zeros(M, N);
    for i = 1:numel(index_row)
        Features = zeros(testNum, 9*16+59);
        u = 1;
        for j = 1:numel(index_col)
            Box = Box1 ( index_row(i)-ww : index_row(i)+ww , ...
                index_col(j)-ww : index_col(j)+ww );
                Blocks{1} = Box(1:7  , 1:7);
                Blocks{2} = Box(6:12 , 1:7);
                Blocks{3} = Box(11:17, 1:7);
                Blocks{4} = Box(1:7  , 6:12);
                Blocks{5} = Box(6:12 , 6:12);
                Blocks{6} = Box(11:17, 6:12);
                Blocks{7} = Box(1:7  , 11:17);
                Blocks{8} = Box(6:12 , 11:17);
                Blocks{9} = Box(11:17, 11:17);
            t = 0;
            for k = 1:N_block
                h = hist (Blocks{k}(:), 0:15);
                Features (u, t+1:t+16) = h;
                t = t + 16;
            end
            Box = Box2 ( index_row(i)-ww : index_row(i)+ww , ...
                index_col(j)-ww : index_col(j)+ww );
```

```
        h = hist (Box(:), 0:58);
        Features (u, 145:end) = h;
        % ------------------------------------------------------------------------------------------
        u = u + 1;
        % ------------------------------------------------------------------------------------------
    end
    %
==============================================================
    oldFolder = cd('F:/My m.files/Thesis/Algorithms/Classification');
    % ------------------------------------------------------------------------------------------
%       FF = Features;
    % ------------------------------------------------------------------------------------------
    Features = Features / sum(Features(1,:));
    Features  = (Features-repmat(Min, [testNum 1])) ./ repmat(Max-Min, [testNum 1]);
    if ~isempty(A)
        Features = A*(Features'-repmat(miu',1,testNum));
        Features = Features';
        d = size(Model.SVs, 2);
        Features = Features(:,1:d);
    end
    % ------------------------------------------------------------------------------------------
    %     DM_test = zeros(testNum, trainNum);
    %     DM_tmp = zeros(testNum, trainNum);
    %     for jj = 1:1
    %         % ------------------------------------------------------------------------------
    %         for ii = 1:testNum
    %             test = Features(ii,:,jj);
    %             %             DM_tmp2(ii,:) = Distance_measure(trains(:,:,jj), test)';
    %             DM_tmp(ii,:) = sum (trains(:,:,jj) .* test(ones(1,trainNum),:), 2)';
    %         end
    %         % ------------------------------------------------------------------------------
    %         %         DM_tmp2 = DM_tmp2.^1 * Weight(jj);
    %         DM_test = DM_test + DM_tmp;
    %     end
```

```
        % ------------------------------------------------------------------------------------
        cd('libsvm-3.1/matlab');
        Kernel_test = Features;%[(1:testNum)' (1+DM_test).^2];
        [predict_label, ~, decision_values] = svmpredict(ClassIDs, Kernel_test, Model);
        index = find( predict_label );
        % ------------------------------------------------------------------------------------
%       OO = [OO ; FF(index,:)];
        % ------------------------------------------------------------------------------------
        gcf;
        if ~isempty(index)
            plot (cah,index_col(index)+1, index_row(i)+1, 'sy')
            pause(0.01)
        end
        % ------------------------------------------------------------------------------------
        Label{N_scale} (index_row(i)+1, index_col+1) = predict_label;
        decision_matrix{N_scale} (index_row(i)+1, index_col+1) = -decision_values;
        % ------------------------------------------------------------------------------------
        cd(oldFolder)
    end
    %
=============================================================
    output = repmat(image, [1 1 3]);
    [Ix1 Ix2] = find( Label{N_scale} );
    for i = 1:numel(Ix1)
        row1 = Ix1(i)-2*w;
        row2 = Ix1(i)+2*w;
        if Ix1(i) <= 2*w
            row1 = 1;
        end
        if Ix1(i) > M-2*w
            row2 = M;
        end
        col1 = Ix2(i)-2*w;
        col2 = Ix2(i)+2*w;
```

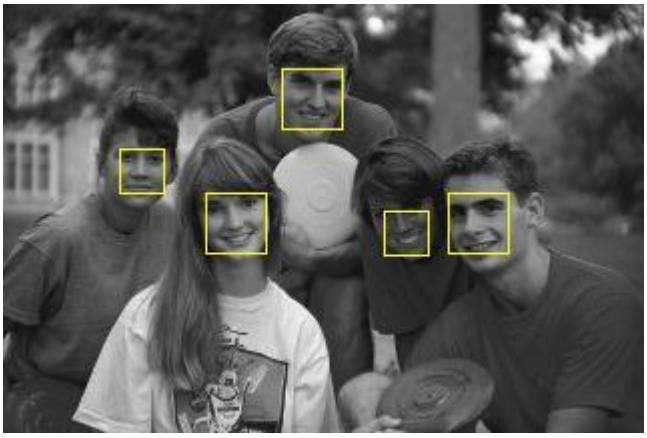
```
        if Ix2(i) <= 2*w
            col1 = 1;
        end
        if Ix2(i) > N-2*w
            col2 = N;
        end
        % ------------------------------------------------------------------------------------------
        block = Label{N_scale} (Ix1(i)-w:Ix1(i)+w , Ix2(i)-w:Ix2(i)+w);
        if sum(sum(block)) > 1
            block = decision_matrix{N_scale} (row1:row2 , col1:col2);
            flag = max(block(:));
            if decision_matrix{N_scale} (Ix1(i), Ix2(i)) == flag
                output (Ix1(i)-w:Ix1(i)+w, [Ix2(i)-w Ix2(i)+w], 3) = 0;
                output (Ix1(i)-w:Ix1(i)+w, [Ix2(i)-w Ix2(i)+w], [1 2]) = 255;
                output ([Ix1(i)-w Ix1(i)+w], Ix2(i)-w:Ix2(i)+w, 3) = 0;
                output ([Ix1(i)-w Ix1(i)+w], Ix2(i)-w:Ix2(i)+w, [1 2]) = 255;
                % ------------------------------------------------------------------------------
                index = round( (([Ix1(i) Ix2(i)]-0.5) * scaling_ratio^(N_scale-1)) + 0.5 );
                Final_DecValue(index(1),index(2)) = Final_DecValue(index(1),index(2)) +
flag;
                Final_Label(index(1),index(2)) = Final_Label(index(1),index(2)) + 1;
                if flag > Max_DecValue(index(1),index(2))
                    Max_DecValue(index(1),index(2)) = flag;
                    Final_Scale(index(1),index(2)) = N_scale;
                end
            else
                %               Face_Pos(i,:) = 0;
            end
        else
            %           Face_Pos(i,:) = 0;
        end
        % ------------------------------------------------------------------------------------------
    end
    close(gcf)
```

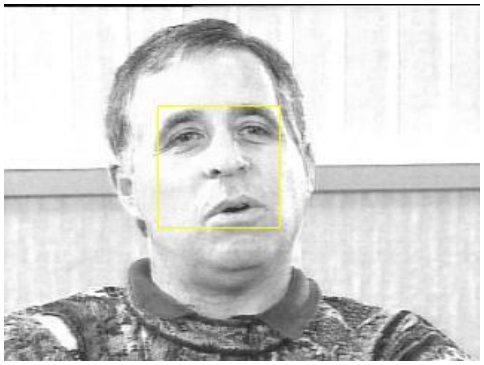
```
    if mod(N_scale,2)== 1
        figure; imshow(uint8(output));
        title(['Scale ',num2str(N_scale)], 'FontSize', 12)
    end
end
% Face_Pos(Face_Pos==0) = [];
% ------------------------------------------------------------------------------------------------
------
[row col val] = find(Final_DecValue);
sub = (col-1)*siz1 + row;
[val ind] = sort(val, 'descend');
row = row(ind);
col = col(ind);
sub = sub(ind);
side = round (window/2 * scaling_ratio.^(Final_Scale(sub)-1));
bound = round (2 * scaling_ratio.^(Final_Scale(sub)-1));
bound(mod(bound,2)==1) = bound(mod(bound,2)==1) + 1;
temp = rem(side, 2);
side(~temp) = side(~temp) - 1;
% ------------------------------------------------------------------------------------------------
-
flag1 = 1;
flag2 = 2;
while flag1 <= numel(row)
    block = Final_Label(row(flag1)-bound(flag1):row(flag1)+bound(flag1) , ...
        col(flag1)-bound(flag1):col(flag1)+bound(flag1));
    if sum(sum(block)) < 2
        row(flag1) = [];
        col(flag1) = [];
        side(flag1) = [];
        bound(flag1) = [];
    else
        while flag2 <= numel(row)
            if (abs(row(flag1)-row(flag2)) > side(flag1)+side(flag2)) || ...
```

```
                (abs(col(flag1)-col(flag2)) > side(flag1)+side(flag2))
            flag2 = flag2 + 1;
        else
            % ------------------------------------------------------------------------------------------
---
            row(flag2) = [];
            col(flag2) = [];
            side(flag2) = [];
            bound(flag2) = [];
        end
    end
    flag1 = flag1 + 1;
    flag2 = flag1 + 1;
  end
end
% ------------------------------------------------------------------------------------------------
------
output = repmat(I, [1 1 3]);
for i = 1:numel(row)
   output (row(i)-side(i):row(i)+side(i), [col(i)-side(i) col(i)+side(i)], 3) = 0;
   output (row(i)-side(i):row(i)+side(i), [col(i)-side(i) col(i)+side(i)], [1 2]) = 255;
   output ([row(i)-side(i) row(i)+side(i)], col(i)-side(i):col(i)+side(i), 3) = 0;
   output ([row(i)-side(i) row(i)+side(i)], col(i)-side(i):col(i)+side(i), [1 2]) = 255;
end
close all
figure; imshow(uint8(output));
title('Final Refined Result', 'FontSize', 12)
% ------------------------------------------------------------------------
time = toc;
```

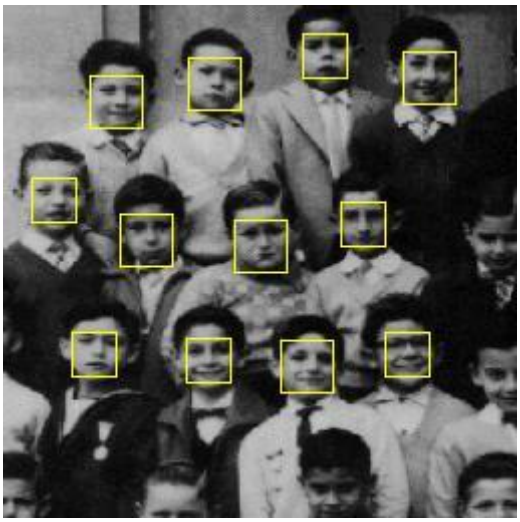
۵-۴- نتایج شبیه‌سازی

شکل ۵-۲ نتیجه شبیه‌سازی ۱

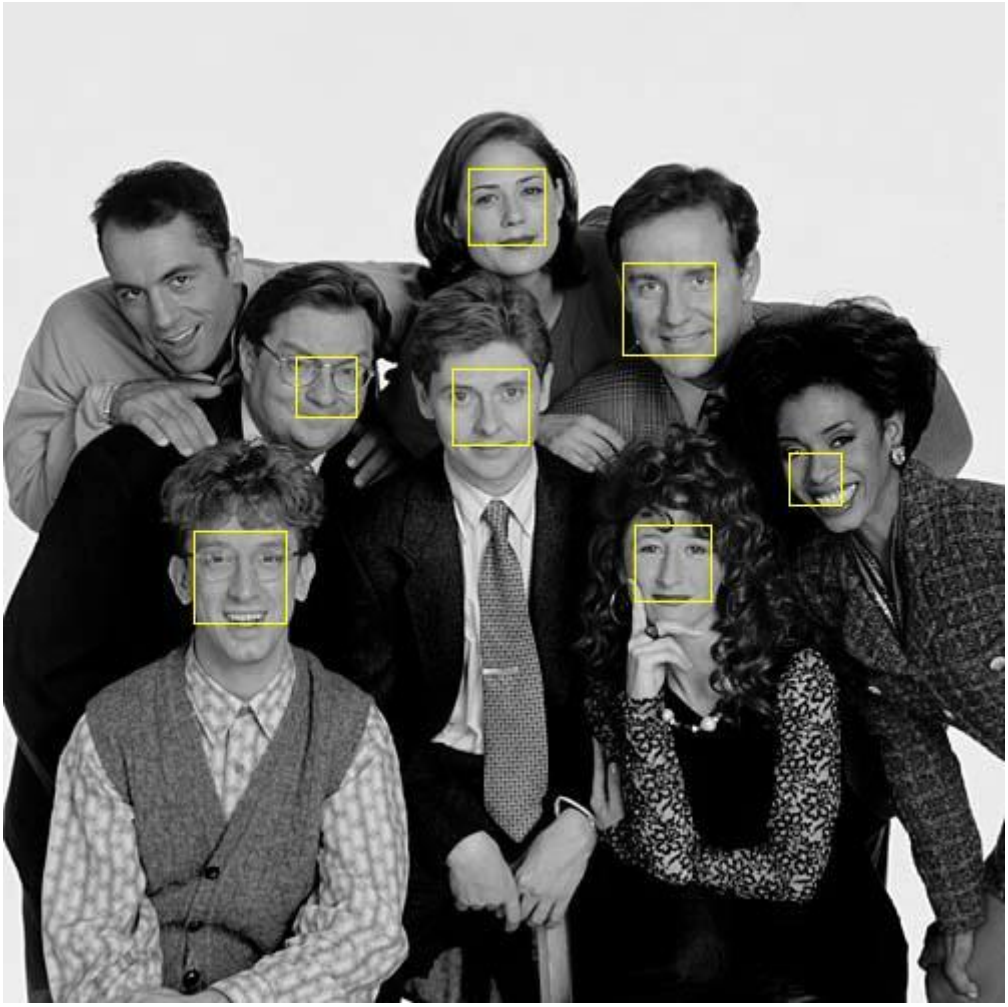




شکل ۵-۳ نتیجه شبیه‌سازی ۲

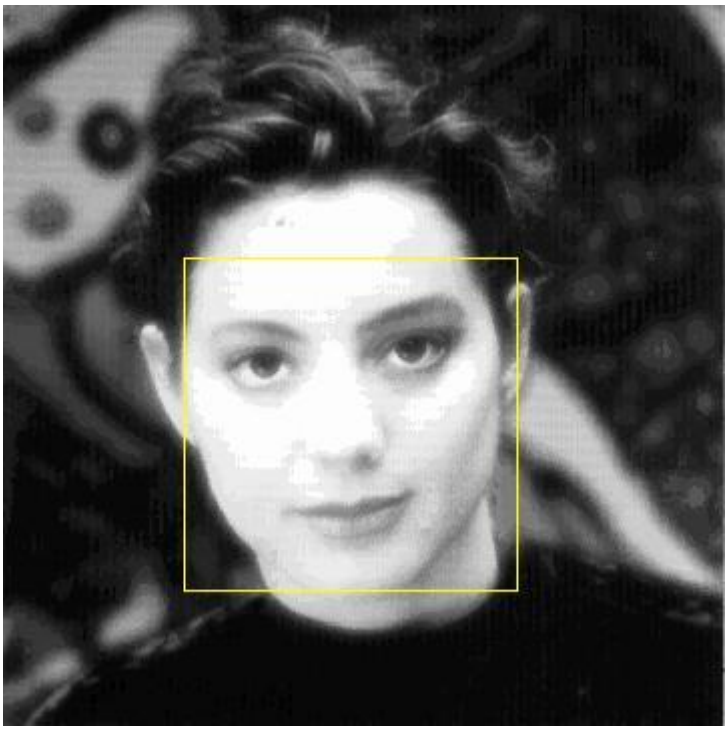


شکل ۵-۴ نتیجه شبیه‌سازی ۳



شکل ۵-۵ نتیجه شبیه‌سازی ۴

شکل ۵-۶ نتیجه شبیه‌سازی ۵



شکل ۵-۷ نتیجه شبیه‌سازی ۶

مراجع

[1] T.Ahonen, A.Hadid & M.Pietikنinen, Face Description with Local Binary Patterns: Application to Face Recognition. Draft, 5th June 2006.
[2] A.Hadid, M.Heikkilن, T.Ahonen & M.Pietikنinen. A Novel Approach to Access Control based on Face Recognition. Machine Vision Group, InfoTech Oulu and Department of Electrical and Information Engineering. University of Oulu, Finland.
[3] T.Ahonen, M.Pietikنinen, A.Hadid & T.Mنenpaن. Face Recognition Based on the Appearance of Local Regions. Machine Vision Group, InfoTech. University of Oulu, Finland. 2004 IEEE.
[4] T.Ahonen, A.Hadid & M.Pietikنinen. Face Recognition with Local Binary Patterns. Machine Vision Group, InfoTech. University of Oulu, Finland. T.Pajdla and J.Matas (Eds): ECCV 2004, LNCS 3021, pp.469-481, 2004. Spring-Verlag Berlin Heidelberg 2004.
[5] A.Hadid & M.Pietikنinen. A Hybrid Approach to Face Detection under Unconstrained Environments. Machine Vision Group, Dept. of Electrical and Information Engineering. University of Oulu, Finland.
[6] A.Hadid, M.Pietikنinen & T.Ahonen. A Discriminative Feature Space for Detecting and Recognizing Faces. Machine Vision Group, InfoTech Oulu and Dept. of Electrical and Information Engineering. University of Oulu, Finland.
[7] T.Ojala, M.Pietikنinen & T.Mنenpaن. Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971-987, July 2002.
[8] G.Heusch, Y.Rodriguez & S.Marcel. Local Binary Patterns as an Image Preprocessing for Face Authentication. IDIAP Research Institute, Martigny, Switzerland. Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
[9] Y.Rodriguez & S.Marcel. Face Authentication using Adapted Local Binary Pattern Histograms. IDIAP Research Institute, Martigny, Switzerland.
[10] S.Marcel, Y.Rodriguez & G.Heusch. On the Recent Use of Local Binary Patterns for
Face Authentication. IDIAP Research Institute, Martigny, Switzerland. International Journal of Image and Video Processing, Special Issue on Facial Image Processing. May 2007.
[11] E.Osuna, R.Freund & F.Girosi. Training Support Vector Machine: an Application to
Face Detection. In Proc. Computer Vision and Pattern Recognition, pages 130-136, June 1997.
[12] Josep R. Casas, F. Marqués & P.Salembier. Apunts de l'assignatura: Processament d'Imatge. Image Processing Group, Signal Theory & Comm. Dept, UPC. Barcelona, Fall 2004.
[13] Lindsay I Smith. A tutorial on Principal Components Analysis. February 26, 2002.
[14] K.-K. Sung, Learning and Example Selection for Object and Pattern Detection, PhD
thesis, MIT AI Lab, Jan. 1996. Available as AI Technical Report 1572.
[15] G.Yang & T.S.Huang. Human Face Detection in Complex Background, Pattern Recognition, vol. 27, no. 1, pp. 53-63, 1994.
[16] K.C.Yow & R.Cipolla. Feature-based human face detection, Image and Vision Computing, vol. 15, no. 9, pp. 713 – 735, 1997.
[17] A.L.Yuille, P.W.Hallinan & D.S.Cohen. Feature extraction from faces using

deformable templates, International Journal of Computer Vision vol. 8, no. 2, 99–111, 1992.
[18] P.Juell & R.Marsh. A hierarchical neural network for human face detection, Pattern Recog. 29, 1996, 781–787.
[19] E.Viennet & F.Fogelman Soulié. Connectionist methods for human face processing, in
Face Recognition: From Theory to Application, Springer-Verlag, Berlin/New York, 1998.
[20] M.Schulze, K.Scheffler, & K.W.Omlin. Recognizing Facial Actions with Support Vector Machines, In Proc. PRASA, pp. 93-96, 2002.
[21] L.R.Rabiner. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings IEEE, vol. 77, no. 2, Feb 1989.
[22] M.A.Turk & A.P.Pentland. Face recognition using eigenfaces, Proceedings of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp. 586-591, Maui, Hawaii 1991.