در تابع Gabor\_Mean ابتدا تمام عکس های منفی و مثبت را به سایز 100\*100 برده و روی آن ها گبر میزنیم.برای هر تصویر مثبت یا منفی یک بردار به دست می آید که نهایتا در این تابع میانگین بردار های مثبت و منفی را جداگانه حساب کرده و درون فایل میریزیم.

سپس روی تصاویر test یک for میزنیم و تمام تصاویر را روی تابع similarityCalculator اعمال میکنیم.در این تابع ابتدا یک پنجره ی 40\*100 روی تصویر جدا میشود.این پنجره را (currentWindow) به تابع testGabor میدهیم، در این تابع مثل تابع Gabor\_Mean این بار برای تصاویر test گبر را اعمال میکنیم.به این ترتیب که یک پنجره ی 40\*100 را به عنوان ورودی دریافت کرده آن را 100\*100 میکند و روی آن گبر میزند.

سپس به تابع similarityCalculator برمیگردیم و پنجره ی 40\*100 را 5 پیکسل جلوتر منتقل میکنیم و توصیفگر را به دست می آوریم. و به همین ترتیب تا انتهای تصویر جلو میرویم.نهایتا با اعمال فاصله ی اقلیدسی روی توصیف گر ها یک درصد شباهت با تصاویر مثبت و یک در صد تشابه با تصاویر منفی به دست می آوریم.(similarity, disSimilarity)

با تقسیم این دو عدد بر هم ، یک ماتریس به دست می آید که برای هر پنجره ی 40\*100 تصویر میزان شباهت یا عدم شباهت آن قسمت با ماشین را مشخص میکند.

سپس در تابع findMaxWindow تصویر و ماتریس شباهت ها و عدم شباهت ها را دریافت میکنیم و بیشترین عدد (که بیانگر شبیه ترین پنجره است) را به دست می آوریم.آن پنجره را به عنوان ماشین در تصویر در نظر میگیریم و دور آن مستطیل میکشیم. در اینجا از الگوریتم های دیگری مثل local max یا قرار دادن یک حد آستانه میتوان استفاده کرد.

نهایتا تمام تصاویر test را با همین الگوریتم پردازش کرده و آن ها را در فولدر FinalImages سیو میکنیم.

\*به دلیل کمبود وقت دقت الگوریتم دقت بالایی نیست ☹ و فقط از ماکسیمم گرفتن بین کل ویندو ها تصاویر حاصل را به دست می آورد. اگر وقت بیشتری بود local max و روش های بهتری را هم میشد استفاده کرد که دقت بالاتر و بهتری داشته باشند D: