
بسم الله الرحمن الرحيم

دانشکده فنی و مهندسی –گروه مهندسی برق

عنوان:

طراحی و پیاده‌سازی سیستم کنترل حرکات ربات مبتنی بر ردیابی حرکات چشم

نویسنده:

سینا فیضی

استاد راهنما:

دکتر حسن مرادزاده

پاییز ۱۴۰۳

چکیده

این پروژه به طراحی و پیاده‌سازی سیستمی برای **ردیابی دقیق حرکات چشم** مبتنی بر پردازش تصویر و الگوریتم‌های یادگیری ماشینی پرداخته است. این سیستم به‌گونه‌ای طراحی شده که بتواند موقعیت چشم را تشخیص داده و اطلاعات مرتبط با موقعیت آن را برای کنترل دستگاه‌ها و سیستم‌های دیگر ارسال کند. این فناوری در حوزه‌هایی نظیر کنترل ربات، توانبخشی افراد ناتوان و کاربردهای پزشکی، می‌تواند راه‌حل‌های موثری ارائه دهد. مهم‌ترین نوآوری این سیستم استفاده از **فیلتر کالمن** به‌عنوان روشی برای کاهش نویز در داده‌های استخراج‌شده از شمارش پیکسل‌های تیره و کمک به بهبود دقت و پایداری در تشخیص موقعیت چشم است. با استفاده از فیلتر کالمن، داده‌های ناپایدار اولیه که تحت تاثیر نویزها و حرکات ناخواسته قرار دارند به سیگنالی هموار و پایدار تبدیل شده‌اند.

در این پروژه، برای بهبود وضوح تصویر و تشخیص بهتر ناحیه چشم، از تکنیک‌های مختلف پردازش تصویر استفاده شده است. این تکنیک‌ها شامل ایجاد ماسک برای جداسازی ناحیه چشم، اعمال فیلترهای میانگین‌گیر و هموارسازی و تبدیل به مقیاس خاکستری هستند. پس از استخراج ناحیه چشم و اعمال پیش‌پردازش‌های مختلف، از **فیلتر کالمن** برای پیش‌بینی و تصحیح داده‌ها استفاده شده است که با پیش‌بینی مقادیر بعدی و تصحیح آن‌ها، تاثیر نویزهای احتمالی را به حداقل رسانده و موجب بهبود پایداری سیستم شده است. این فیلتر همچنین امکان تنظیم دقیق‌تر بر اساس شرایط محیطی مختلف را فراهم کرده و سیستم را در برابر تغییرات نور و حرکات نامطلوب مقاوم ساخته است.

نتایج نشان می‌دهد که سیستم با استفاده از فیلتر کالمن، عملکردی پایدارتر و دقیق‌تر نسبت به روش‌های بدون این فیلتر دارد ﷺ**کاربردهای این سیستم بسیار گسترده‌اند** و در حوزه‌هایی مانند کنترل‌های کمکی برای افراد ناتوان، سیستم‌های آموزشی مبتنی بر ردیابی چشم، مانیتورینگ روانشناسی و کنترل صنعتی و رباتیک کاربرد دارد. همچنین، به‌دلیل کاهش قابل‌توجه نویز و افزایش دقت، این سیستم قابلیت استفاده در شرایط محیطی چالش‌برانگیز را نیز دارد.

پیشنهاد می‌شود در آینده با بهره‌گیری از **روش‌های یادگیری عمیق** و الگوریتم‌های پیشرفته‌تر پردازش تصویر، دقت این سیستم بهبود یابد و قابلیت‌های بیشتری به آن اضافه شود. این سیستم می‌تواند به‌عنوان یک ابزار استاندارد و دقیق برای ردیابی و تحلیل رفتار چشم در حوزه‌های مختلف مورد استفاده قرار گیرد و در دسترس محققان و صنایع مختلف قرار گیرد.

کلمات‌کلیدی: تشخیص حرکات چشم، تشخیص‌چهره، کنترل ربات با چشم، هوش‌مصنوعی، یادگیری‌عمیق

فهرست

فصل اول

مقدمه

۱–۱–مقدمه

با گسترش فناوری و پیشرفت‌های چشمگیر در زمینه هوش مصنوعی، رباتیک و پردازش تصویر، راه‌های جدیدی برای تعامل انسان با ماشین پدیدآمده است. در این میان، روش‌های طبیعی‌تر برای کنترل ربات‌ها و دستگاه‌های الکترونیکی روزبه‌روز اهمیت بیشتری پیدا می‌کنند، زیرا این روش‌ها می‌توانند نیاز به استفاده از ابزارهای فیزیکی را کاهش داده و تجربه کاربری را بهبود بخشند. یکی از این روش‌های نوآورانه، کنترل حرکات ربات از طریق ردیابی حرکات چشم است. این فناوری به کاربران اجازه می‌دهد تا تنها با حرکت چشم خود بتوانند یک ربات را به سمت‌های مختلف هدایت کرده و وظایف مختلفی را انجام دهند. چنین سیستمی پتانسیل کاربردهای گسترده‌ای در حوزه‌های مختلف از جمله پزشکی، توان‌بخشی، صنعت و حتی سرگرمی دارد.

کنترل ربات توسط حرکات چشم می‌تواند به‌ویژه برای افرادی که با محدودیت‌های حرکتی مواجه هستند، مانند افراد دارای ناتوانی‌های جسمی یا حرکتی، بسیار مفید باشد. این فناوری به آن‌ها امکان می‌دهد بدون نیاز به ابزارهای دستی یا حرکات فیزیکی پیچیده، با ربات‌ها و دستگاه‌های اطراف خود تعامل کنند. همچنین، این روش می‌تواند در محیط‌های صنعتی و شرایطی که امکان استفاده از دست‌ها یا ابزارهای دستی وجود ندارد، بسیار کارآمد باشد و به بهبود بهره‌وری و ایمنی کمک کند.

به‌منظور تحقق این هدف، در این پژوهش سعی شده است تا از فناوری‌های پردازش تصویر و یادگیری ماشین برای شناسایی و تحلیل حرکات چشم استفاده شود. به کمک الگوریتم‌های پیشرفته ردیابی چشم و تفسیر حرکات، سیستم می‌تواند جهت نگاه و تغییرات حرکتی چشم را به دستورات کنترل ربات تبدیل کند. برای مثال، درصورتی‌که کاربر به سمت راست نگاه کند، ربات به همان سمت حرکت می‌کند و در صورت نگاه به جهت مخالف، واکنش متناسبی از سوی ربات به دست می‌آید. این فرایند، نیازمند دقت و سرعت بالا در پردازش داده‌های تصویری است تا بتواند واکنش‌های آنی و بدون تأخیر ارائه دهد.

در بخش‌های مختلف این پژوهش، چالش‌های فنی از جمله‌پردازش سریع تصاویر، تشخیص دقیق حرکات چشم، فیلترکردن نویزها و عوامل مزاحم و همچنین بهینه‌سازی الگوریتم‌ها برای عملکرد بی‌درنگ بررسی شده است. این سیستم باید توانایی شناسایی و دنبال‌کردن حرکات چشم کاربر را در شرایط نوری و محیطی متفاوت داشته باشد تا در کاربردهای مختلف، نتایج دقیق و پایدار ارائه دهد.

۱-۲- ضرورت و اهداف تحقیق

پژوهش حاضر به بررسی کاربردهای عملی این فناوری در شرایط واقعی پرداخته و امکان پیاده‌سازی این سیستم در موقعیت‌های مختلف، بررسی می‌شود. هدف از این تحقیق، طراحی سیستمی است که نه‌تنها دقت و کارایی بالایی داشته باشد، بلکه استفاده از آن نیز برای کاربران ساده و راحت باشد. این تلاش می‌تواند گامی مؤثر در جهت توسعه فناوری‌های تعامل انسان و ربات باشد که بر پایه رفتارهای طبیعی انسان بنا شده‌اند و پتانسیل ایجاد انقلابی در حوزه‌های مختلف را دارند.

۱-۲-۱ اهدف تحقیق

۱. **طراحی و توسعه یک سیستم کنترل ربات مبتنی بر ردیابی حرکات چشم:** هدف اصلی تحقیق، طراحی و پیاده‌سازی سیستمی است که بتواند حرکات چشم کاربر را به دستورات حرکتی برای ربات تبدیل کند. این سیستم باید به‌گونه‌ای باشد که بادقت بالا و تأخیر کم، جهت نگاه و حرکات چشم کاربر را شناسایی و به حرکات ربات تبدیل کند.

۲. **بررسی و تحلیل الگوریتم‌های پردازش تصویر و یادگیری ماشین برای ردیابی حرکات چشم:** در این تحقیق، انواع الگوریتم‌های پردازش تصویر و یادگیری ماشین جهت شناسایی و ردیابی حرکات چشم بررسی و تحلیل می‌شوند تا مناسب‌ترین روش‌ها برای پیاده‌سازی سیستم انتخاب شوند. هدف این است که این الگوریتم‌ها دقت و پایداری بالایی داشته و بتوانند در شرایط نوری مختلف عملکرد مناسبی ارائه دهند.

۳. **ارزیابی دقت و سرعت عملکرد سیستم در شرایط واقعی:** برای کاربرد عملی این سیستم، آزمایش‌هایی در شرایط نوری و محیطی متفاوت انجام می‌شود تا دقت و سرعت پاسخ‌دهی سیستم به حرکات چشم کاربر سنجیده شود. هدف این بخش، تضمین عملکرد مناسب سیستم در شرایط واقعی و غیر آزمایشگاهی است.

۴. **توسعه یک رابط کاربری ساده و کاربرپسند:** به‌منظور بهبود تجربه کاربری، طراحی رابط کاربری ساده و قابل‌فهم برای کاربران در نظر گرفته شده است. هدف این است که کاربران بتوانند به‌راحتی با سیستم ارتباط برقرار کرده و از آن استفاده کنند، به‌ویژه افرادی که دارای محدودیت‌های حرکتی هستند.

۵. **شناسایی و حل چالش‌های فنی و بهینه‌سازی عملکرد سیستم:** در جریان این تحقیق، چالش‌های فنی از جمله دقت در شناسایی حرکات چشم، حذف نویزها و عوامل مزاحم، و پردازش سریع اطلاعات بررسی و برای آن‌ها راهکارهایی ارائه می‌شود. هدف این است که سیستم به سطحی از عملکرد برسد که بتواند به‌صورت بی‌درنگ و بادقت بالا عمل کند.

فصل دوم

پیشینه تحقیق

۲-۱- پیشینه تحقیق

در زمینه کنترل ربات‌ها با استفاده از حرکات چشم، پژوهش‌های مختلفی انجام شده است که به بهبود تعامل انسان و ربات کمک کرده‌اند. استفاده از فناوری ردیابی چشم به دلیل ماهیت طبیعی و غیرتهاجمی آن، موردتوجه بسیاری از محققان بوده و به‌ویژه در حوزه‌های پزشکی و توان‌بخشی، نقش مهمی ایفا کرده است. پیشینه این تحقیق بر اساس مطالعات قبلی و پیشرفت‌های علمی در این حوزه به شرح زیر ارائه می‌شود.

در یکی از تحقیقات پایه‌ای، سیستم‌های ردیابی چشم با استفاده از روش‌های پردازش تصویر معرفی شده‌اند که امکان شناسایی و دنبال‌کردن حرکات چشم را در زمان واقعی فراهم می‌کنند [۱]. این سیستم‌ها از الگوریتم‌های پردازش تصویر و یادگیری ماشین بهره می‌برند تا با دقت و سرعت بالایی حرکات چشم را تحلیل کنند.

مطالعات دیگری نیز نشان داده‌اند که ردیابی حرکات چشم برای کنترل ربات‌ها در کاربردهای پزشکی به‌ویژه در توان‌بخشی بیماران با محدودیت‌های حرکتی، مؤثر بوده است. در تحقیقی، از ردیابی چشم برای کنترل یک بازوی رباتیک استفاده شده و نتایج نشان داده است که این فناوری به طور مؤثری می‌تواند جایگزین کنترل‌های دستی در محیط‌های خاص شود [۲].

پژوهشگران به بهینه‌سازی الگوریتم‌های تشخیص و ردیابی حرکات چشم پرداخته‌اند تا این سیستم‌ها بادقت بالاتری حتی در شرایط نوری مختلف کار کنند. مطالعه‌ای با بررسی الگوریتم‌های مختلف پردازش تصویر نشان داد که ترکیب فیلترهای تطبیقی و یادگیری عمیق می‌تواند باعث افزایش دقت و کاهش نویز در سیستم‌های ردیابی چشم شود [۳].

همچنین، در برخی تحقیقات به بررسی کاربردهای این فناوری در محیط‌های صنعتی پرداخته شده است که در آن‌ها استفاده از حرکات چشم برای کنترل ربات‌ها در محیط‌های سخت و غیرقابل‌دسترس برای اپراتور انسان مورد بررسی قرار گرفته است [٤]. این تحقیقات به اهمیت و نقش این فناوری در افزایش ایمنی و بهره‌وری در محیط‌های صنعتی تاکید دارند.

علاوه بر این، تحقیقاتی به بررسی طراحی رابط کاربری ساده و کاربرپسند برای سیستم‌های ردیابی چشم پرداخته‌اند. هدف از این پژوهش‌ها، ایجاد رابط‌هایی است که برای کاربران، به‌ویژه افرادی که دارای

محدودیت‌های حرکتی هستند، به‌راحتی قابل‌استفاده باشند [۵]. این امر به بهبود تجربه کاربری و پذیرش بهتر این فناوری در کاربردهای روزمره کمک کرده است.

در سال‌های اخیر، ترکیب فناوری‌های تشخیص چهره و ردیابی چشم برای کنترل ربات‌ها، توجه بسیاری از محققان را به خود جلب کرده است. سیستم‌های تشخیص چهره با استفاده از پردازش تصویر و یادگیری ماشین قادر به شناسایی و دنبال‌کردن چهره کاربر هستند که این امر در تعاملات هوشمند بین انسان و ربات نقش مهمی دارد. این ترکیب فناوری‌ها به‌ویژه در توسعه رابط‌های کاربری غیر لمسی که به کاربران امکان کنترل ربات‌ها را از طریق حرکات صورت و چشم می‌دهند، اهمیت ویژه‌ای پیدا کرده است.

در پژوهشی که به تشخیص چهره و کنترل ربات از طریق آن پرداخته است، از الگوریتم‌های یادگیری عمیق برای شناسایی چهره و هدایت ربات استفاده شده است. این تحقیق نشان می‌دهد که ترکیب تشخیص چهره و ردیابی چشم می‌تواند دقت و پایداری بیشتری برای کنترل ربات در محیط‌های پیچیده فراهم کند [٦].

مطالعه دیگری نشان داده است که استفاده از تکنیک‌های تشخیص چهره در کنار ردیابی چشم می‌تواند در بهبود تجربه کاربری برای افرادی که با محدودیت‌های حرکتی مواجه هستند، مؤثر باشد. در این تحقیق، چهره و جهت نگاه کاربر شناسایی شده و به کمک آن‌ها دستورات حرکتی به ربات ارسال شده است [۷]. این فناوری به کاربران اجازه می‌دهد تا با حرکات ساده سر و چشم خود، ربات را به سمت‌ها و جهات مختلف هدایت کنند.

همچنین تحقیقات نشان داده است که سیستم‌های ترکیبی تشخیص چهره و ردیابی چشم می‌توانند در شرایط نوری متغیر نیز عملکرد مناسبی داشته باشند. در یکی از این مطالعات، از ترکیب روش‌های پردازش تصویر و یادگیری عمیق برای مقابله با چالش‌های ناشی از تغییرات نور و زاویه استفاده شده است. نتایج نشان داده است که این سیستم‌ها به‌ویژه در محیط‌های پیچیده و صنعتی که نورپردازی به‌خوبی کنترل نشده است، عملکرد مؤثری دارند [۸].

پژوهش دیگری به بررسی کاربردهای این سیستم در حوزه‌های امنیتی و نظارت پرداخته است. این تحقیق نشان داده است که استفاده از تشخیص چهره و ردیابی حرکات چشم می‌تواند به سیستم‌های امنیتی هوشمند و ربات‌های نگهبان امکان شناسایی و پیگیری چهره‌های مشکوک را بدهد. این فناوری می‌تواند به افزایش ایمنی در محیط‌های حساس کمک کند و به‌عنوان یک ابزار قدرتمند برای نظارت هوشمند مورداستفاده قرار گیرد [۹].

همچنین، در برخی تحقیقات به بررسی کارایی سیستم‌های ترکیبی در تعاملات اجتماعی و نقش آن‌ها در بهبود تعامل انسان و ربات پرداخته شده است. برای مثال، در مطالعه‌ای که بر تعاملات اجتماعی متمرکز است، استفاده از تشخیص چهره و ردیابی چشم به ربات‌ها امکان می‌دهد تا با شناسایی حرکات صورت و چشم، به واکنش‌های هوشمندانه‌ای پاسخ دهند و تعامل طبیعی‌تری با انسان داشته باشند [۱۰]. این فناوری به بهبود ارتباط و تعامل میان انسان و ربات کمک کرده و می‌تواند در ربات‌های دستیار خانگی و همراهان اجتماعی کاربرد داشته باشد.

فصل سوم

طراحی و ساخت

۳–۱– مقدمه

در این فصل، به طراحی و پیاده‌سازی سیستم تشخیص و ردیابی چهره می‌پردازیم. این سیستم با استفاده از OpenCV، Dlib و مجموعه‌ای از روش‌های پردازش تصویر و فیلتر کالمن پیاده‌سازی شده است. همچنین این سیستم به یک میکروکنترلر متصل است که وظیفه کنترل سروو موتورها و نمایش چشم‌ها را دارد. این سیستم شامل دو بخش اصلی پردازش تصویر و سخت‌افزار کنترل‌کننده است.

۳–۲– طرح پیشنهادی

سیستم پیشنهادی شامل دو بخش اصلی نرم‌افزاری و سخت‌افزاری است که به‌صورت زیر قابل‌توصیف هستند:

۱. **بخش نرم‌افزاری:**

تشخیص چهره و نقاط کلیدی صورت با استفاده از کتابخانه‌های OpenCV و Dlib ، سیستم قادر به شناسایی چهره و استخراج نقاط کلیدی صورت (مانند چشم‌ها) کاربر است.

ردیابی حرکات چشم: با تحلیل نقاط استخراج شده و استفاده از الگوریتم‌های پردازش تصویر و فیلتر کالمن، موقعیت نگاه چشم‌ها (چپ، مرکز، راست) تعیین می‌شود.

ارسال داده‌ها به میکروکنترلر: موقعیت تشخیص‌داده‌شده از طریق wifi به میکروکنترلر ارسال می‌شود تا دستورات لازم برای حرکت سروو موتورها صادر گردد.

۲. **بخش سخت‌افزاری:**

میکروکنترلر :در این پروژه از میکروکنترلرESP32 به علت پشتیبانی از wifi جهت ارتباط با کامپیوتر و کنترل موتور سروو استفاده شده است.

سروو موتورها: برای حرکت‌دادن بخش‌های مختلف ربات استفاده می‌شوند.

نمایشگرOLED : برای نمایش چشم‌ها به‌صورت گرافیکی و ارائه بازخورد بصری به کاربر استفاده می‌شود که در این پروژه از جهت زیباسازی استفاده شده که می‌تواند در آینده برای

ارتقای عملکرد وضعیت واقعی چشم‌ها را به نمایش بگذارد یا یک دوربین با آن جایگزین شود و بتوان اطراف را مانند چشم و بافاصله از طریق اینترنت مشاهده کرد. همچنین رگولاتورهایی برای تغذیه میکروکنترلر و موتورها تعبیه شده‌اند.

۳-۳- الگوریتم تشخیص حرکت چشم

ابتدا تصویر وب‌کم سیستم دریافت شده و به رنگ خاکستری تبدیل می‌شود. سپس از الگوریتم شناسایی چهره Dlib استفاده می‌شود تا چهره شناسایی شود. اگر چهره‌ای در تصویر پیدا شود، آن را با یک مستطیل سبزرنگ بر روی تصویر رسم می‌کند. برای شناسایی ویژگی‌های صورت (مثل چشم‌ها، بینی، لب‌ها و غیره)، از مدل *"shape_predictor_68_face_landmarks.dat"*[1] استفاده می‌شود که ٦٨ نقطه مختلف از صورت را شناسایی می‌کند. سپس، نقاط مربوط به چشم‌ها استخراج شده و برای ردیابی حرکات چشم‌ها مورداستفاده قرار می‌گیرند.

تصویر چشم‌ها به منطقه‌ای کوچک برش داده می‌شود تا پردازش روی آن محدود به همین بخش شود. سپس از تکنیک‌های مختلف پردازش تصویر برای استخراج ویژگی‌های مهم استفاده می‌شود.

۳-۳-۱ پردازش تصویر

مرحله تشخیص چشم‌ها و پردازش تصویر برای ردیابی شامل چندین تکنیک پردازش تصویر است که هر کدام به بهبود کیفیت و دقت شناسایی موقعیت چشم کمک می‌کنند. این بخش از الگوریتم به چندین مرحله تقسیم می‌شود تا در نهایت بتوانیم موقعیت چشم‌ها را بادقت بیشتری تعیین کنیم. در ادامه، مراحل مختلف این بخش را توضیح داده خواهد شد.

۱. استخراج ناحیه چشم

- ابتدا با استفاده از نقاط ویژگی چهره که در مرحله قبلی شناسایی شده‌اند، ناحیه چشم از تصویر خاکستری برش داده می‌شود. نقاطی که مربوط به هر چشم هستند، به‌عنوان نقاطی برای تعریف مرزهای چشم استفاده می‌شوند.

[1] یک مدل آماده برای تشخیص اجزای صورت است.

- این برش تصویر باعث می‌شود تا تنها ناحیه مربوط به چشم برای پردازش در نظر گرفته شود که باعث افزایش سرعت پردازش و کاهش نویز می‌شود.

۲. بهبود کنتراست تصویر با استفاده ازCLAHE

- **CLAHE (Contrast Limited Adaptive Histogram Equalization)** یک تکنیک پیشرفته برای بهبود کنتراست در تصویرهای خاکستری است. این روش به بهبود کنتراست در مناطق کوچک تصویر کمک می‌کند و باعث می‌شود ویژگی‌های چشم مانند مردمک و لبه‌ها واضح‌تر شوند.
- این کار به شناسایی بهتر مرزهای چشم کمک می‌کند، زیرا تفاوت بین ناحیه‌های روشن و تیره (مثل مردمک و سفیدی چشم) در تصویر افزایش می‌یابد.

۳. ایجاد ماسک برای محدودکردن پردازش به ناحیه چشم

- با استفاده از نقاط مرزی چشم، یک ماسک برای چشم ایجاد می‌شود. این ماسک به شکل یک چندضلعی است که دقیقاً روی چشم قرار می‌گیرد.
- با استفاده از این ماسک، تنها قسمت داخل چندضلعی به‌عنوان چشم شناخته می‌شود و سایر بخش‌ها حذف می‌شوند. سپس [1]عملیات بیتی برای ترکیب تصویر اصلی و ماسک اعمال می‌شود تا فقط ناحیه چشم باقی بماند.
- این ماسک‌گذاری به حذف نویزهای خارجی کمک می‌کند و باعث تمرکز پردازش بر روی ناحیه دقیق چشم می‌شود.

۴. تبدیل تصویر به فضای رنگی HSV و تشخیص پیکسل‌های تاریک

- تصویر اصلی به فضای رنگی **HSV (Hue, Saturation, Value)** تبدیل می‌شود. در این فضا، مقدار روشنایی برای تشخیص نقاط تیره (مثل مردمک) استفاده می‌شود.
- سپس، محدوده رنگی برای تشخیص پیکسل‌های سیاه و تاریک (پیکسل‌های داخل مردمک) تنظیم می‌شود و یک ماسک دیگر ساخته می‌شود که تنها پیکسل‌های تیره را نگه می‌دارد.
- این مرحله به کاهش نور پس‌زمینه و جداسازی مردمک از سایر بخش‌های چشم کمک می‌کند.

۵. اعمال فیلترهای صاف‌کننده و کاهش نویز

- **فیلتر میانگین‌گیری (Median Blur)** و **فیلتر گوسین (Gaussian Blur)** برای کاهش نویزهای ریز در تصویر اعمال می‌شوند.

[1] bitwise AND

- فیلتر میانگین‌گیری به‌خصوص برای حذف نقاط ریز و تیز بسیار مفید است، زیرا پیکسل‌های منفردی که ممکن است به‌عنوان نویز شناخته شوند را حذف می‌کند.
- این کار باعث می‌شود تا تصویر صاف‌تر و آماده‌تری برای پردازش بیشتر داشته باشیم و لبه‌های موردنظر بهتر شناسایی شوند.

۶. تنظیم روشنایی و کنتراست ناحیه چشم

- با استفاده از تابع **convertScaleAbs**، میزان روشنایی و کنتراست تصویر تنظیم می‌شود. این کار با دو پارامتر انجام می‌شود alpha :ضریب کنتراست و beta : روشنایی.
- این تنظیمات باعث می‌شود ناحیه‌های تیره‌تر (مثل مردمک) و ناحیه‌های روشن‌تر (مثل سفیدی چشم) بیشتر برجسته شوند، که در نتیجه به تشخیص بهتر موقعیت مردمک کمک می‌کند.

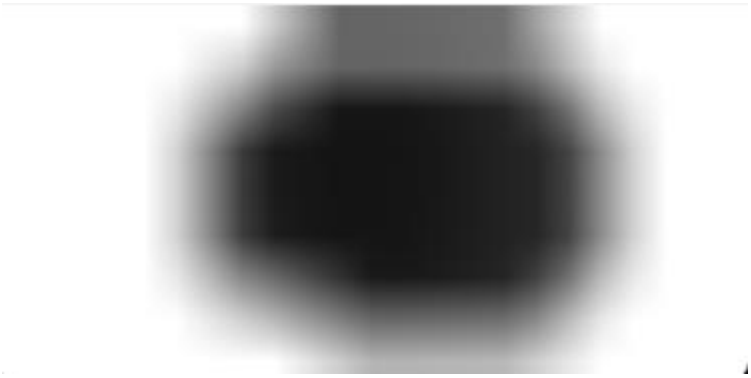
۷. استخراج و برش نهایی ناحیه چشم

- پس از پردازش تصویر و بهبود کیفیت آن، چشم به بخش‌های کوچک‌تری تقسیم می‌شود. تصویر ناحیه چشم به سه بخش **چپ**، **مرکز** و **راست** تقسیم شده و در هر بخش تعداد پیکسل‌های سیاه شمارش می‌شود.
- این شمارش پیکسل‌های سیاه در هر بخش به‌عنوان یک ویژگی کلیدی برای تعیین موقعیت چشم (چپ، مرکز، راست) استفاده می‌شود.

۸. آستانه‌گذاری(Thresholding)

- در نهایت، آستانه‌گذاری بر روی ناحیه چشم اعمال می‌شود تا بخش‌های تاریک و روشن به‌صورت باینری (۰ و ۲۵۵) جدا شوند. این کار کمک می‌کند تا مردمک و سفیدی چشم واضح‌تر از هم جدا شوند و قسمت‌های کاملا تیره به‌عنوان مردمک شناخته شوند
- این مرحله برای تحلیل پیکسل‌ها در هر بخش از ناحیه چشم ضروری است، زیرا حالا پیکسل‌های تاریک مردمک و پیکسل‌های روشن‌تر سفیدی چشم به‌خوبی از هم متمایز شده‌اند.

پس از انجام این مراحل تصویری که از چشم‌ها به دست می‌آید در شکل ۳-۲-۱ نمایش داه شده است.



شکل۳-۲-۱

در شکل ۳-۲-۱ تصویر به دست آمده از چشم را با انجام مراحل بالا مشاهده می‌شود که به راحتی می توان پیکسل‌های مشکی تصویر را در تصویر تشخیصداد.

در الگوریتم این پروژه این تصویر به سه قسمت مساوی تقسیم می‌شود و در نهایت قسمتی که تعداد پیکسل‌های مشکی آن بیشتر است به‌عنوان خروجی در نظر گرفته می‌شود که هم در صفحه‌نمایش داده می‌شود و هم جهت ربات را مشخص می‌کند.

شکل ۳-۲-۲ تصویر چشم را به صورت سیاه سفید و بدون استفاده از ماسک و فیلترها نشان می‌دهد.

شکل-۳-۲-۲

در این پروژه چندین ماسک و فیلتر برای پردازش تصویر استفاده شده‌اند که به بهبود دقت و وضوح در تشخیص موقعیت چشم کمک می‌کنند. در ادامه، هر کدام از ماسک‌ها و فیلترهای استفاده شده و تأثیرات آن‌ها به‌تفصیل توضیح داده می‌شود:

۱. ماسک چندضلعی(Polygon Mask)

این ماسک با استفاده از نقاط مرزی دور چشم ایجاد می‌شود و به شکل یک چندضلعی است که دقیقاً روی ناحیه چشم قرار می‌گیرد.

این ماسک باعث می‌شود تا تنها ناحیه دقیق چشم پردازش شود و بقیه نواحی تصویر حذف شوند. به‌این‌ترتیب، نویزها و اشیا خارجی که ممکن است در اطراف چشم باشند، تأثیری بر نتیجه نهایی ندارند. این تمرکز بر ناحیه چشم، باعث افزایش سرعت پردازش و دقت تشخیص مردمک می‌شود.

۲. ماسک سیاه(Black Mask)

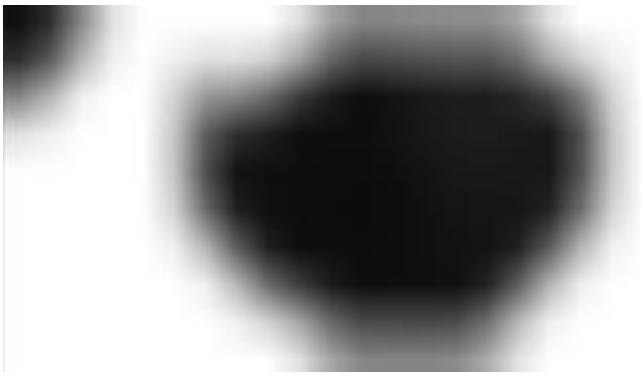
این ماسک بر اساس تبدیل تصویر به فضای رنگی HSV و شناسایی پیکسل‌های تاریک در محدوده رنگی سیاه (مردمک) ایجاد می‌شود. این ماسک به‌خوبی مردمک چشم را از سفیدی چشم و بقیه بخش‌ها جدا می‌کند. با استفاده از این ماسک، می‌توان دقیقاً موقعیت مردمک را مشخص کرد و نویزهای ناشی از روشنایی یا انعکاس‌های نور در ناحیه چشم را کاهش داد. شکل ۳-۳ تصویر چشم را بدون ماسک و شکل ۳-۲-٤ با ماسک را نشان می‌دهند.

شکل۳-۳

شکل۳-۲-٤

٤. فیلتر گوسین(Gaussian Blur)

این فیلتر با ترکیب‌کردن پیکسل‌های اطراف هر نقطه، تصویر را صاف‌تر و یکنواخت‌تر می‌کند. فیلتر گوسین باعث کاهش نویز و نقاط تیز ناخواسته می‌شود. این کار به شناسایی مرزهای نرم‌تر و طبیعی‌تر کمک می‌کند و مانع از ثبت جزئیات غیرضروری می‌شود. به همین دلیل، استفاده از این فیلتر به جلوگیری از شناسایی اشتباه مردمک و بهبود کیفیت نهایی تصویر کمک می‌کند. شکل ۳-٤ تصویر را بدون فیلتر گوسین و شکل ۳-۵ تصویر را با فیلتر گوسین نشان می‌دهند.

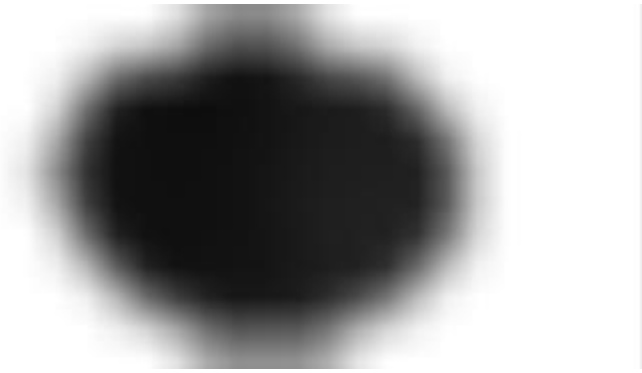


شکل-۳-۲-۵

۵. فیلتر میانگین‌گیری(Median Blur)

فیلتر میانگین‌گیری یکی از فیلترهای صاف‌سازی است که هر پیکسل را با مقدار میانگین پیکسل‌های اطراف آن جایگزین می‌کند.

این فیلتر برای کاهش نویزهای تیز و لکه‌های کوچک استفاده می‌شود. با استفاده از این فیلتر، پیکسل‌های منفرد که ممکن است به‌عنوان نویز شناخته شوند، حذف می‌شوند. این کار به بهبود دقت تشخیص مردمک و جلوگیری از اشتباهات در شناسایی ناحیه تاریک مردمک کمک می‌کند. شکل۳-٦ نیز تصویر را با فیلتر میانگین‌گیری نشان می‌دهد.



شکل۳-۲-٦

۷. فیلتر تنظیم روشنایی و کنتراست(ConvertScaleAbs)

این فیلتر از پارامترهای alpha و beta برای تنظیم کنتراست و روشنایی تصویر استفاده می‌کند. با تغییر این پارامترها، تفاوت روشنایی بین ناحیه تاریک مردمک و ناحیه روشن سفیدی چشم بیشتر می‌شود. این کار به شناسایی دقیق‌تر مرز مردمک کمک کرده و باعث افزایش دقت در تعیین موقعیت مردمک می‌شود.

۳-۳-۲ فیلتر کالمن

فیلتر کالمن یک تکنیک پردازشی ریاضی است که برای تخمین مقدار یک متغیر در زمانی خاص استفاده می‌شود و بر اساس داده‌های گذشته و اندازه‌گیری‌های فعلی بهبود می‌یابد. هدف اصلی این فیلتر، کاهش نویز و بهینه‌سازی داده‌ها در شرایطی است که داده‌های ورودی ممکن است با خطا و نویز همراه باشند.

در این پروژه، فیلتر کالمن برای تخمین موقعیت چشم و پردازش بهتر داده‌های مربوط به پیکسل‌های تاریک مردمک در نواحی مختلف چشم (چپ، راست، و مرکز) استفاده شده است. فیلتر کالمن به کمک داده‌های فعلی (تعداد پیکسل‌های تاریک هر بخش) و تخمین‌های گذشته، مقدارهای دقیق‌تری را محاسبه می‌کند. این فرایند به هموارسازی داده‌ها کمک کرده و باعث می‌شود که نوسانات ناگهانی و نویزها کمتر بر نتایج تأثیر بگذارند.

مفهوم کلی فیلتر کالمن

فیلتر کالمن بر اساس یک مدل دومرحله‌ای عمل می‌کند:

پیش‌بینی: در این مرحله، فیلتر کالمن بر اساس مقادیر قبلی داده‌ها یک تخمین اولیه برای وضعیت فعلی تولید می‌کند.

به‌روزرسانی: سپس، مقدار تخمینی به کمک داده‌های اندازه‌گیری‌شده به‌روزرسانی می‌شود. این به‌روزرسانی به شکلی انجام می‌شود که خطای تخمین به حداقل برسد.

این دو مرحله به‌صورت تکراری در هر بار ورودی داده جدید اجرا می‌شوند، و فیلتر کالمن همواره در حال بهبود تخمین‌ها و کاهش نویز است.

پارامترهای فیلتر کالمن در پروژه

فیلتر کالمن مورداستفاده دارای سه پارامتر اصلی است:

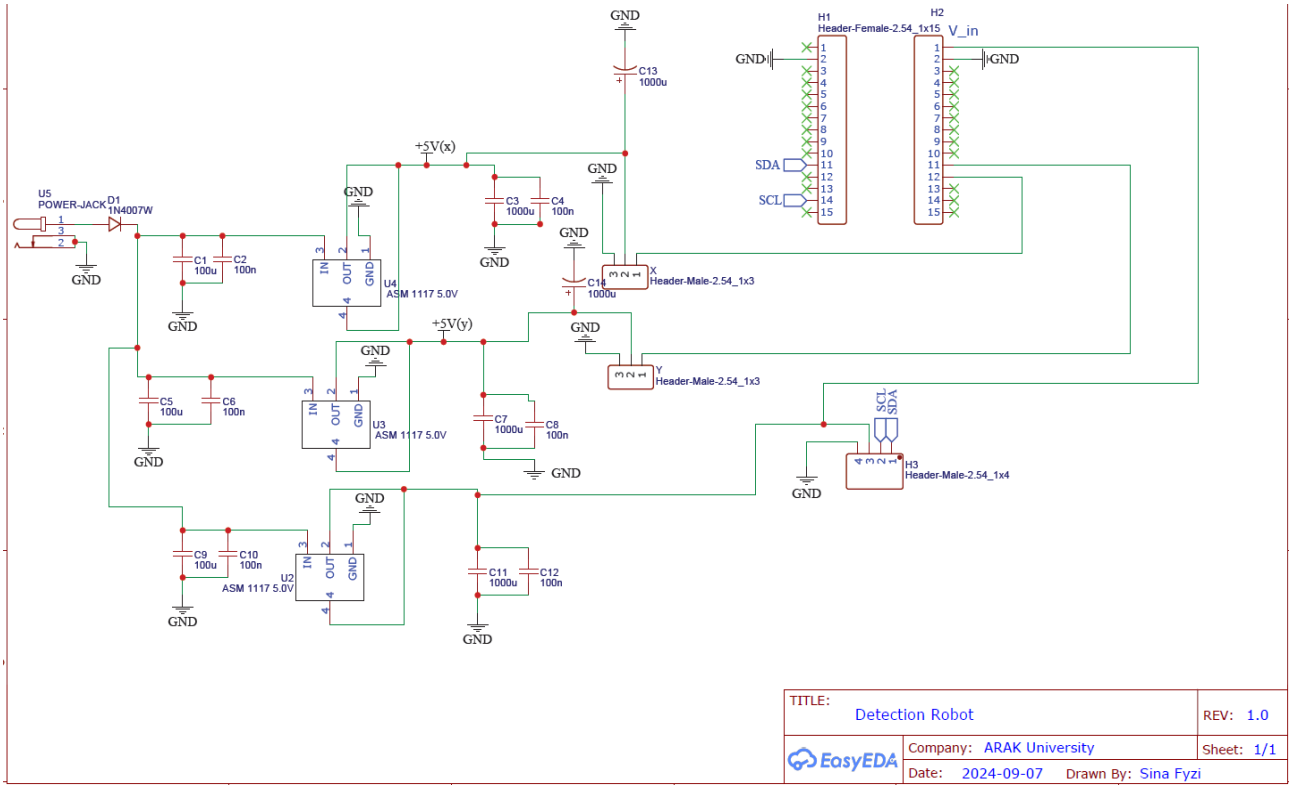
واریانس فرآیند: این پارامتر مقدار نویز موجود در فرآیند را نشان می‌دهد و معمولاً به‌عنوان میزان اطمینان از مدل دینامیکی فرآیند شناخته می‌شود. در این پروژه مقدار واریانس فرآیند برابر با ۰.۰۱ در نظر گرفته شده است. مقدار پایین این پارامتر نشان‌دهنده اعتماد بالای فیلتر به مدل دینامیکی تخمین شده است و فیلتر تنها تغییرات کوچک را می‌پذیرد.

واریانس اندازه‌گیری: این پارامتر مقدار نویز یا عدم قطعیت در داده‌های ورودی را نشان می‌دهد. در این پروژه، مقدار واریانس اندازه‌گیری برابر ۲ است. این مقدار نشان‌دهنده اطمینان نسبی به داده‌های ورودی است به این معنا که داده‌ها ممکن است نویز داشته باشند؛ اما همچنان معتبر در نظر گرفته می‌شوند.

مقدار اندازه‌گیری اولیه: این پارامتر مقدار اولیه تخمین زده‌شده از تعداد پیکسل‌های تاریک است. در پروژه، این مقدار برابر با صفر تنظیم شده است. فیلتر از این مقدار به‌عنوان نقطه شروع استفاده می‌کند و در طی پردازش، مقدار آن را با داده‌های ورودی به‌روزرسانی می‌کند.

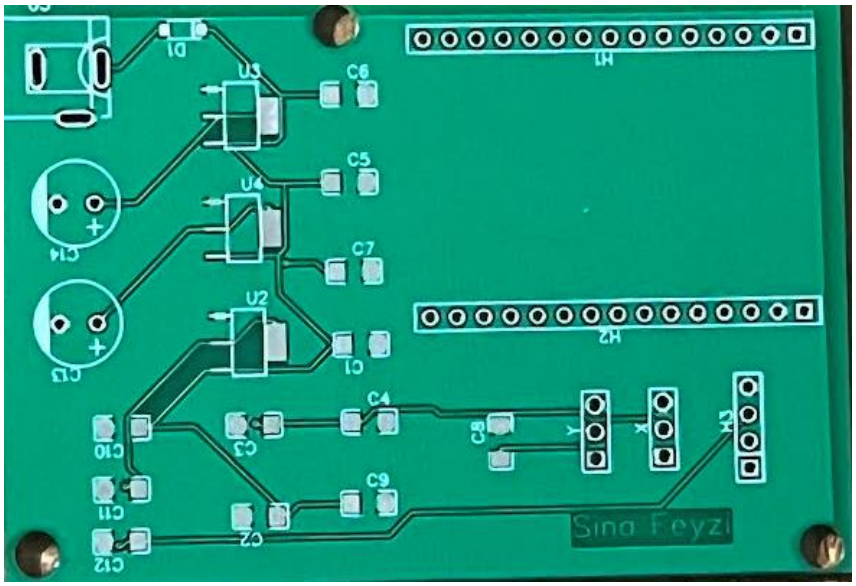
۳-۳-۳ سخت‌افزار ربات

سخت‌افزار این پروژه شامل دو سروو موتور، یک ESP32، یک صفحه‌نمایش گرافیکی و برد مدارچاپی که برای قرارگیری مدار و المان‌های آن طراحی شده و از سه رگولاتور ولتاژ ۵ ولتی برای تغذیه ربان استفاده شده است. شکل ۳-۳-۱ شماتیک برد را نشان می‌دهد.



شکل ۳-۳-۱

شکل۳-۳-۲ نیز برد ساخته شده را نشان می‌دهد.



شکل۳-۳-۲

قطعات به کار برده شده در پروژه نیز به شرح جدول ۳-۳-۱ است.

جدول۳-۳-۱: قطعات برد چاپی

Name	Designator	Footprint	Quantity	Column2
100u	**C1,C5,C9**	C1206		
100n	**C2,C4,C6,C8,C10,C12**	C1206	6	
1000u	**C3,C7,C11**	C1206	3	
1000u	**C13,C14**	CAP-D8.0×F3.5	2	
1N4007W	**D1**		1	1N4007W
Header-Female-2.54_1x15	**H1,H2**		2	B-2200S15P-A120
Header-Male-2.54_1x4	**H3**		1	B-2100S04P-A110
ASM 1117 5.0V	**U2,U3,U4**	ASM1117 5.0 V	3	
POWER-JACK	**U5**		1	POWER-JACK
Header-Male-2.54_1x3	**X,Y**		2	

۳-۳-۴ نتیجه‌گیری

در این پروژه، سیستمی برای ردیابی دقیق چشم‌ها بر پایه پردازش تصویر و یادگیری ماشینی پیاده‌سازی شد که قادر است حرکات چشم را تشخیص داده و اطلاعات مربوط به موقعیت چشم را برای کنترل سایر دستگاه‌ها و سیستم‌ها منتقل کند. ویژگی نوآورانه اصلی این سیستم، استفاده از فیلتر کالمن به عنوان ابزاری برای بهبود دقت و کاهش نویز در داده‌های مربوط به شمارش پیکسل‌های تیره است که به طور مستقیم از تصویر چشم استخراج می‌شوند. این فیلتر نقش کلیدی در پایدارسازی داده‌ها و حذف خطاهای احتمالی ناشی از حرکات ناخواسته و لرزش‌های کوچک دارد و دقت در تشخیص موقعیت چشم را به‌طور چشم‌گیری افزایش می‌دهد.

یکی از چالش‌های اصلی در ردیابی چشم، تداخل نویزهای محیطی و تغییرات کوچک در چهره یا دوربین است که موجب می‌شود تشخیص درست موقعیت چشم دشوار گردد. در این پروژه، فیلتر کالمن با **تعدیل و تثبیت مقادیر شمارش پیکسل‌های تیره** در هر بخش از چشم به کمک روش‌های پیش‌بینی و تصحیح، امکان هموارسازی دقیق‌تری را فراهم کرده است. با این رویکرد، داده‌های مربوط به حرکت چشم به یک سیگنال پایدارتر تبدیل می‌شوند و احتمال بروز خطاهای کاذب در تشخیص موقعیت چشم کاهش می‌یابد. این تکنیک به طور خاص برای موقعیت‌هایی که داده‌ها تحت تاثیر نویزهای محیطی و حرکات کوچک ناخواسته قرار دارند بسیار موثر و نوآورانه بوده است.

مزایا و نوآوری‌های کلیدی در استفاده از فیلتر کالمن و سایر فیلترها

۱. کاهش نویز و هموارسازی دقیق داده‌ها: فیلتر کالمن در این پروژه به‌طور خاص برای حذف نویز در داده‌های مربوط به تعداد پیکسل‌های تیره استفاده شده است، که نتیجه آن بهبود دقت تشخیص موقعیت چشم‌ها است.

۲. پیش‌بینی موقعیت‌های احتمالی بر اساس داده‌های قبلی**: یکی از ویژگی‌های فیلتر کالمن، پیش‌بینی مقادیر بعدی بر اساس داده‌های گذشته است. این پیش‌بینی موجب کاهش تاخیر در پاسخ‌دهی سیستم و فراهم‌سازی حرکتی روان‌تر و پایدارتر می‌شود.

۳. انعطاف‌پذیری در مواجهه با شرایط محیطی متغیر: استفاده از این فیلتر سیستم را در برابر تغییرات نور محیط و حرکات ناخواسته مقاوم‌تر می‌کند و باعث می‌شود حتی در شرایط محیطی نامناسب نیز تشخیص موقعیت چشم با دقت بالا انجام شود.

کاربردهای آینده و توسعه‌های ممکن

۱. سیستم‌های کنترلی برای افراد معلول: با بهره‌گیری از فیلتر کالمن و به حداقل رساندن نویزهای حرکتی، می‌توان سیستم‌هایی را توسعه داد که با حرکات دقیق چشم کار کنند و برای افراد دارای ناتوانی‌های حرکتی قابل اعتماد باشند.

۲. کاربرد در حوزه‌های پزشکی و روان‌شناسی: سیستم‌های ردیابی چشم که بتوانند به دقت رفتار چشم را در شرایط مختلف تحلیل کنند می‌توانند در آزمایش‌ها و تحقیقات مربوط به روان‌شناسی و پزشکی مفید واقع شوند.

۳. **کنترل رباتیک و کاربردهای صنعتی**: استفاده از فیلتر کالمن در ردیابی چشم، سیستم‌هایی پایدار برای کنترل ربات‌ها در شرایط صنعتی ایجاد می‌کند و امکان مانیتورینگ یا کنترل از راه دور را فراهم می‌کند.

نتیجه‌گیری نهایی

این پروژه با به‌کارگیری فیلتر کالمن به عنوان هسته اصلی در کاهش نویز داده‌ها و بهبود پایداری سیستم، گام بلندی در راستای افزایش دقت ردیابی چشم برداشته است. این تکنیک نه تنها دقت بالاتری را ارائه می‌دهد، بلکه با کاهش اثرات منفی نویز و بهبود سیگنال‌های خروجی، سیستمی کارا و پایدار برای کاربردهای بلادرنگ ایجاد کرده است. با توسعه و بهینه‌سازی‌های بیشتر، این سیستم می‌تواند به یکی از ابزارهای استاندارد در زمینه‌هایی چون توانبخشی، رباتیک و آموزش تبدیل شود. همچنین فیلترهای پرادزش تصویر باعث شده تا تصاویری ملموس برای پیاده‌سازی الگوریتم‌ها در اختیار داشته باشیم.

در آینده، استفاده از الگوریتم‌های یادگیری عمیق و روش‌های پیشرفته‌تری برای پردازش و تحلیل تصاویر چشم، می‌تواند دقت و قابلیت‌های این سیستم را بهبود بخشد و زمینه را برای استفاده گسترده‌تر در حوزه‌های گوناگون فراهم کند.

پیوست

کد پروژه در پایتون:

```
import cv2 as cv
import numpy as np
import dlib
import math
import time
import requests

fonts = cv.FONT_HERSHEY_COMPLEX
YELLOW = (0, 247, 255)
CYAN = (255, 255, 0)
MAGENTA = (255, 0, 242)
GREEN = (0, 255, 0)
LIGHT_CYAN = (255, 204, 0)
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
url = "http://192.168.4.1"

detectFace = dlib.get_frontal_face_detector()

predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")


class KalmanFilter:
    def __init__(self, process_variance, measurement_variance,
estimated_measurement):
        self.process_variance = process_variance
        self.measurement_variance = measurement_variance
        self.estimated_measurement = estimated_measurement
        self.error_covariance = 1.0

    def update(self, measurement):
        # Kalman gain
        kalman_gain = self.error_covariance / (self.error_covariance +
self.measurement_variance)
        # Update estimated measurement
        self.estimated_measurement = self.estimated_measurement + kalman_gain *
(measurement - self.estimated_measurement)
        # Update error covariance
```

```
        self.error_covariance = (1 - kalman_gain) * self.error_covariance +
self.process_variance
        return self.estimated_measurement



kf_right = KalmanFilter(process_variance=1e-2, measurement_variance=1,
estimated_measurement=0)
kf_center = KalmanFilter(process_variance=1e-2, measurement_variance=1,
estimated_measurement=0)
kf_left = KalmanFilter(process_variance=1e-2, measurement_variance=1,
estimated_measurement=0)


def midpoint(pts1, pts2):
    x, y = pts1
    x1, y1 = pts2
    xOut = int((x + x1) / 2)
    yOut = int((y1 + y) / 2)
    return (xOut, yOut)

def eucaldainDistance(pts1, pts2):
    x, y = pts1
    x1, y1 = pts2
    eucaldainDist = math.sqrt((x1 - x) ** 2 + (y1 - y) ** 2)
    return eucaldainDist

def faceDetector(image, gray, Draw=True):
    cordFace1 = (0, 0)
    cordFace2 = (0, 0)
    faces = detectFace(gray)
    face = None
    for face in faces:
        cordFace1 = (face.left(), face.top())
        cordFace2 = (face.right(), face.bottom())
        if Draw:
            cv.rectangle(image, cordFace1, cordFace2, GREEN, 2)
    return image, face

def faceLandmakDetector(image, gray, face, Draw=True):
    landmarks = predictor(gray, face)
    pointList = []
    for n in range(0, 68):
        point = (landmarks.part(n).x, landmarks.part(n).y)
        pointList.append(point)
```

```
        if Draw:
            cv.circle(image, point, 3, YELLOW, 1)
    return image, pointList


def EyeTracking(image, gray, eyePoints):
    clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    gray = clahe.apply(gray)

    dim = gray.shape
    mask = np.zeros(dim, dtype=np.uint8)
    PollyPoints = np.array(eyePoints, dtype=np.int32)

    cv.polylines(image, [PollyPoints], isClosed=True, color=(0, 255, 0),
thickness=2)
    cv.fillPoly(mask, [PollyPoints], 255)

    eyeImage = cv.bitwise_and(gray, gray, mask=mask)

    hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)

    lower_black = np.array([0, 0, 0])
    upper_black = np.array([180, 255, 65])
    mask_black = cv.inRange(hsv_image, lower_black, upper_black)

    eyeImage[mask_black == 0] = 255
    eyeImage = cv.medianBlur(eyeImage, 5)
    eyeImage = cv.GaussianBlur(eyeImage, (3, 3), 0)
    #eyeImage = cv.bilateralFilter(eyeImage, 9, 75, 75)
    #eyeImage = cv.boxFilter(eyeImage, -1, (5, 5))
    alpha = 1
    beta = -15
    eyeImage = cv.convertScaleAbs(eyeImage, alpha=alpha, beta=beta)

    maxX = (max(eyePoints, key=lambda item: item[0]))[0]
    minX = (min(eyePoints, key=lambda item: item[0]))[0]
    maxY = (max(eyePoints, key=lambda item: item[1]))[1]
    minY = (min(eyePoints, key=lambda item: item[1]))[1]

    cropedEye = eyeImage[minY+2:maxY, minX+3:maxX-3]

    # Check if cropedEye is valid
    if cropedEye.size == 0:
        print("Cropped eye is empty, skipping this frame.")
        return None, "Eye Closed"
```

```
    height, width = cropedEye.shape
    divPart = int(width / 3)

    _, thresholdEye = cv.threshold(cropedEye, 0, 255, cv.THRESH_BINARY +
cv.THRESH_OTSU)

    rightPart = thresholdEye[0:height, 0:divPart]
    centerPart = thresholdEye[0:height, divPart:divPart + divPart]
    leftPart = thresholdEye[0:height, divPart + divPart:width]

    rightBlackPx = np.sum(rightPart == 0)
    centerBlackPx = np.sum(centerPart == 0)
    leftBlackPx = np.sum(leftPart == 0)

    # Apply Kalman Filter to smooth the pixel values
    rightBlackPx = kf_right.update(rightBlackPx)
    centerBlackPx = kf_center.update(centerBlackPx)
    leftBlackPx = kf_left.update(leftBlackPx)

    pos = Position([int(rightBlackPx), int(centerBlackPx), int(leftBlackPx)])
    #time.sleep(0.02)

    cropped_eye_resized = cv.resize(cropedEye, (300, 150))
    cv.imshow("Cropped Eye", cropped_eye_resized)

    return mask, pos



def Position(ValuesList):
    print(f"Black pixel count: Right={ValuesList[0]}, Center={ValuesList[1]},
Left={ValuesList[2]}")

    maxIndex = ValuesList.index(max(ValuesList))
    posEye = ''

    if maxIndex == 0:
        posEye = "Right"
    elif maxIndex == 1:
        posEye = "Center"
        if ValuesList[1] - ValuesList[0] <= 15 and ValuesList[0]/10>=1:
            posEye = "Right"
        if ValuesList[1] - ValuesList[2] <= 15 and ValuesList[2]/10>=1:
            posEye = "Left"
```

```
    elif maxIndex == 2:
        posEye = "Left"
    else:
        posEye = "Eye Closed"

    return posEye

# Main loop to capture video
cap = cv.VideoCapture(0)
while True:

    ret, frame = cap.read()
    if ret == False:
        break


    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

    frame, face = faceDetector(frame, gray)
    if face is not None:
        frame, pointList = faceLandmakDetector(frame, gray, face)
        leftEyePts = pointList[36:42]
        rightEyePts = pointList[42:48]
        _, leftPos = EyeTracking(frame, gray, leftEyePts)
       # _, rightPos = EyeTracking(frame, gray, rightEyePts)

        cv.putText(frame, f"Left Eye: {leftPos}", (50, 100), fonts, 0.8, GREEN,
2)
        #cv.putText(frame, f"Right Eye: {rightPos}", (50, 150), fonts, 0.7, CYAN,
2)

        cv.imshow("Frame", frame)



        data = ""
        if leftPos == "Left":
            data = "left"
        elif leftPos == "Center":
            data = "center"
        elif leftPos == "Right":
            data = "right"
        print(f"Sending leftPos: {leftPos}, pos: {data}")

        requests.post(f"{url}/control", data= data)
```

```
    key = cv.waitKey(1)


    if key == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

کدپروژه برای ESP32 :

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <WebServer.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
#define SCREEN_ADDRESS 0x3C

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
Servo servoX;

const char *ssid = "SINA_ROBOT";
const char *password = "sina1380";

WebServer server(80);

// Function prototypes
void displayEyes(bool update = true);
void blink(int speed = 32);
void wakeup();
void sleep();
```

```
void move_right_big_eye();
void move_left_big_eye();
void center_eyes();
void handleControl();

int ref_eye_height = 40;
int ref_eye_width = 40;
int ref_space_between_eye = 10;
int ref_corner_radius = 10;

// Current state of the eyes
int left_eye_height = ref_eye_height;
int left_eye_width = ref_eye_width;
int left_eye_x = SCREEN_WIDTH / 2 - ref_eye_width / 2 - ref_space_between_eye /
2;
int left_eye_y = SCREEN_HEIGHT / 2;
int right_eye_x = SCREEN_WIDTH / 2 + ref_eye_width / 2 + ref_space_between_eye /
2;
int right_eye_y = SCREEN_HEIGHT / 2;
int right_eye_height = ref_eye_height;
int right_eye_width = ref_eye_width;

void setup() {
  pinMode(2, OUTPUT);
  servoX.attach(25);
  servoX.write(90);
  Serial.begin(115200);
  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println(F("Sina Feyzi"));
  display.display();
  delay(100);

  WiFi.softAP(ssid, password);
  server.begin();

  server.on("/control", HTTP_POST, handleControl);
```

```
  Serial.println("Access Point Started");

  wakeup();  // Wake up eyes
  center_eyes();
}

void loop() {
  server.handleClient();
  blink(10);
}

void handleControl() {
  if (server.hasArg("plain")) {
    String data = server.arg("plain");


    Serial.println("Received: " + data);


    if (data.equals("left")) {
      servoX.write(-90);
    } else if (data.equals("center")) {
      servoX.write(90);
      //delay(250);
    } else if (data.equals("right")) {
      servoX.write(270);
    }
  }
  server.send(200, "text/plain", "Servos moved");
}


void displayEyes(bool update) {
  display.clearDisplay();

  display.fillRoundRect(left_eye_x - left_eye_width / 2, left_eye_y -
left_eye_height / 2, left_eye_width, left_eye_height, ref_corner_radius,
SSD1306_WHITE);
  display.fillRoundRect(right_eye_x - right_eye_width / 2, right_eye_y -
right_eye_height / 2, right_eye_width, right_eye_height, ref_corner_radius,
SSD1306_WHITE);

  if (update) {
    display.display();
  }
```

```
}

void blink(int speed) {
  for (int i = 0; i < 2; i++) {
    left_eye_height -= speed;
    right_eye_height -= speed;
    displayEyes(true);
    delay(50);
  }
  for (int i = 0; i < 2; i++) {
    left_eye_height += speed;
    right_eye_height += speed;
    displayEyes(true);
    delay(50);
  }
}

void wakeup() {
  for (int h = 2; h <= ref_eye_height; h += 2) {
    left_eye_height = h;
    right_eye_height = h;
    displayEyes(true);
    delay(100);
  }
}

void sleep() {
  left_eye_height = 2;
  right_eye_height = 2;
  displayEyes(true);
}

void move_right_big_eye() {
  for (int w = ref_eye_width; w <= 60; w++) {
    right_eye_width = w;
    displayEyes(true);
    delay(30);
  }
}

void move_left_big_eye() {
  for (int w = ref_eye_width; w <= 60; w++) {
    left_eye_width = w;
    displayEyes(true);
    delay(30);
```

```
  }
}

void center_eyes() {
  left_eye_x = SCREEN_WIDTH / 2 - ref_eye_width / 2 - ref_space_between_eye / 2;
  right_eye_x = SCREEN_WIDTH / 2 + ref_eye_width / 2 + ref_space_between_eye / 2;
  left_eye_width = ref_eye_width;
  right_eye_width = ref_eye_width;
  displayEyes(true);
}
```