ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғұмарбек Дәукеев атындағы Алматы энергетика және байланыс университеті» коммерциялық емес АҚ
Ақпараттық технологиялар институты
«Ақпараттық жүйелер және киберқауіпсіздік кафедрасы» кафедрасы»

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
PhD
_____ А.К. Мукашева
«____» _____ 2021 ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Машиналық оқытуды қолдана отырып, тұлғаны тануға арналған мобильді қосымша құру

Мамандығы: 5В070300– «Ақпараттық жүйелер»

Орындаған: Е.М Сақыбек Тобы: ИСк-17-1
Ғылыми жетекші: PhD, доцент Ф.У Маликова

Кеңесшілер:
Экономикалық бөлім: э.ғ.к., доцент _____Е.М Нұрпейіс
«____»_____2021 ж.

Өміртіршілік қауіпсіздігі: т.ғ.к., доцент _____А.С.Байкенжеева
«____» _____ 2021 ж.

Программалық қамтама бөлімі: PhD, доцент _____ Ф.У Маликова
«____» _____ 2021 ж.

Норма бақылаушы: аға оқытушы _____Б.А. Тулегенова
«____» _____ 2021 ж.

Сын-пікір беруші: ф.-м.ғ.к., доцент _____ Э. Н Нурлыбаева
«____» _____ 2021 ж.

Алматы, 2021

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«Ғұмарбек Дәукеев атындағы Алматы энергетика және байланыс университеті» коммерциялық емес АҚ

Ақпараттық технологиялар институты

«Ақпараттық жүйелер және киберқауіпсіздік кафедрасы» кафедрасы»

Мамандығы 5В070300– «Ақпараттық жүйелер»

Дипломдық жобаны орындауға берілген
ТАПСЫРМА

Білім алушы Сақыбек Ерсұлтан Марданұлы

Жобаның тақырыбы: Машиналық оқытуды қолдана отырып, тұлғаны тануға арналған мобильді қосымша құру
2020 жылғы «27» қазан № 217 университет бұйрығымен бекітілген.

Аяқталған жобаны тапсыру мерзімі: «24» мамыр 2021 ж.

Дипломдық жобаның бастапқы мәліметтері (зерттеу (жоба) нәтижелерінің талап етілген параметрлері мен объектінің бастапқы мәліметтері): Ұсынылып отырған дипломдық жобада Машиналық оқытуды қолдана отырып тұлғаны тануға арналған мобильді қосымша әзірлеу. Жобаны орындау барысында Android SDK плагины және Java тілін қолданамын.

Дипломдық жобада қарастырылған мәселелер тізімі немесе дипломдық жобаның қысқаша мазмұны:
– талдау бөлімі;
– жобалау бөлімі;
– жобаны жүзеге асыру және тестілеу бөлімі;
– экономикалық бөлім;
– өміртіршілік қауіпсіздігі;
– А қосымшасы. Программа листингі;

Графикалық материалдар тізімі (міндетті сызбалар дәл көрсетілуі тиіс): 9 кесте, 35 сурет ұсынылған.

Ұсынылатын негізгі әдебиеттер:
1 Шилдт Г. Java 8. Полное руководство, 9-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 1376 с.
2 Прохоренок Н.А. OpenCV и Java. Обработка изображений и компью-

терное зрение. – СПб.: БХВ – Петербург, 2018. – 320 с.

3 Дэвид А. Форсайт, Жан Понс. Компьютерное зрение. Современный подход, Издательство Вильямс, 2004. – 928 с.

4 Журавлева Л. В., Стригулин К. А. Исследования особенностей развития нейронных сетей в современном мире, Свое издательство, 2016. – 134 с.

5 Прэтт У. Цифровая обработка изображений: Пер.с англ. – М.: Мир, 1982. – Кн. 2 – 480 с.

Дипломдық жобаның бөлімдеріне қатысты белгіленген кеңес берушілер

Бөлімдер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Нұрпейіс Е.М	15.04.2021–30.04.2021	
Өміртіршілік қауіпсіздігі	А.С.Байкенжеева	15.04.2021–30.04.2021	
Программалық қамтама	Ф.У Маликова	13.05.2021-18.05.2021	
Норма бақылау	Б.А. Тулегенова	13.05.2021-18.05.2021	

Дипломдық жобаны дайындау
КЕСТЕСІ

Бөлімдер атауы, арастырылатын мәселелер тізімі	Ғылыми жетекшіге ұсыну мерзімдері	Ескерту
Талдау бөлімі	1.02.2021-12.03.2021	
Жобалау бөлімі	15.03.2021-7.04.2021	
Жобаны жүзеге асыру және тестілеу бөлімі	8.04.2021-20.05.2021	

Тапсырманың берілген күні «___» _____ 20__ ж.

Кафедра меңгерушісі _____ А.К. Мукашева

Жобаның ғылыми жетекшісі _____ Ф.У Маликова

Тапсырманы орындауға алған
білім алушы _____ Е.М Сақыбек

Аңдатпа

Дипломдық жобада қойылған міндеттерге сәйкес, нақты уақыт режимінде тұлғаның бет-әлпетін анықтауға арналған машиналық оқыту технологиясына негізделген мобильді қосымша әзірленді және.

Дипломдық жобаның мақсаты - адам қызметінің әр түрлі салаларында қолдану мүмкіндігі, мысалы, объектіге қол жетімділікті басқару болып табылады.

Дипломдық жобаның соңғы тараулары бағдарламаның аяқталған нұсқасын қолдана отырып, қауіпсіздік бөлімінің әлеуетті қызметкерінің жұмыс орнын және микроклиматтық жағдайын талдауды, сонымен қатар проблемаларға байланысты жағдайларды шешуді қамтитын өмір қауіпсіздігі мәселелерін қарастырылды. Сонымен қатар, осы қосымшаның техникалық-экономикалық негіздемесінің мәселелері қарастырылады, оған жобаны әзірлеуге кететін шығындардың мөлшері есептелді.

Аннотация

В соответствии с задачами, поставленными в дипломном проекте, было разработано мобильное приложение на основе технологии машинного обучения для распознавания лиц в реальном времени .

Цель дипломного проекта - уметь использовать его в различных сферах человеческой деятельности, например, для управления доступом к объекту.

В заключительных главах дипломного проекта были рассмотрены вопросы безопасности жизнедеятельности, в том числе анализ рабочего места и микроклиматического состояния потенциального сотрудника отдела безопасности с использованием завершенной версии программы, а также разрешение ситуаций, связанных с проблемами. Кроме того, рассматриваются вопросы технико-экономического обоснования данного приложения, которое рассчитывается от суммы затрат на разработку проекта.

Annotation

In accordance with the tasks set in the thesis project, a mobile application was developed based on machine learning technology for real-time face recognition and.

The purpose of the thesis project is to be able to use it in various areas of human activity, for example, to control access to an object.

In the final chapters of the diploma project, the issues of life safety were considered, including the analysis of the workplace and the microclimatic state of a potential employee of the security department using the completed version of the program, as well as the resolution of situations related to problems. In addition, the issues of the feasibility study of this application are considered, which is calculated from the amount of costs for the development of the project.

Мазмұны

Кіріспе

Қазіргі әлемде адамды анықтау үшін биометриялық әдістер қолданылады. Биометрия-бұл адамды бірегей физиологиялық немесе мінез-құлық сипаттамаларына негізделген сәйкестендіру әдістері мен құрылғыларының жиынтығы.

Биометриялық әдістер мыналарды қамтиды:

- Саусақ іздерін алу;
- Көз торын анықтау;
- Беттерді;
- Дауысты тану;
- Қолжазбаны тану;

Бұл дипломдық жобаның мақсаты-адамның бет-әлпетін анықтау, сонымен қатар оны нақты уақыт режимінде бақылау функциясын орындайтын қосымшаны іске асыру.

Бетті тану мәселесі интеллектуалды ортада да, қауіпсіздік жүйелерінде де өзекті. Мәселен, биыл Интернетте N-Tech әзірлеген ресейлік технология негізінде фотосуреттер арқылы бет-әлпетті тануға арналған онлайн-қызмет іске қосылды.Lab. Google сонымен қатар стандартты LFW деректер жиынтығында адамның бет-әлпетін 99,63% дәлдікпен танитын жаңа FaceNet жасанды интеллект жүйесі туралы зерттеу жұмысын жариялады. Facebook Researcher жүйесі шамамен 97,5% жинады. Ашық бастапқы өнімдерге OpenCV кіреді. Бұл компьютерлік көру, суретті өңдеу және жалпы мақсаттағы сандық Алгоритмдер кітапханасы. Академиялық және коммерциялық мақсаттарда пайдалану үшін тегін-BSD лицензиясы бойынша таратылады.

Дипломдық жоба кіріспеден, 4 тараудан, қорытындыдан, пайдаланылған көздер тізімінен, қосымшалардан тұрады.

Бірінші тарауда нейрондық желілерді құрудың теориялық негіздері, сондай-ақ бағдарламалық жасақтаманы әзірлеу құралдарын таңдаудың негіздемесі көрсетілген.

Екінші тарауда бағдарламалау технологияларын қолдану, сондай-ақ бағдарламаның бөлігі ретінде компьютерлік көру сипатталған.

Үшінші тарауда бағдарламаны пайдалану кезінде өмір сүру қауіпсіздігін қамтамасыз етуге талдау жасалады.

Төртінші тарауда жобаның техникалық-экономикалық негіздемесі, оны әзірлеу шығындары, сату бағасы және таза пайда есептеледі.

1 Машиналық оқыту технологиясы және мобильді қосымша құру бойынша теориялық негіздеме

1.1 Тұлғаны тануға арналған мобильді қосымша құру бойынша міндет қою

Тапсырма: машинаны оқыту технологиясын қолдана отырып, адамның бет-әлпетін анықтауға және бақылауға арналған мобильді қосымшаны құру.

Жүзеге асыру:

- Әр түрлі ОЖ-ны қолдайтын мобильді қосымша;
- Қосымшамен жұмыс істеу кезінде тіршілік қауіпсіздігі негізгі аспектілерін қарастыру;
- Техникалық-экономикалық негіздемесін және тиімділік қосымшалар.

1.2 Машиналық оқыту технологиясы және тұлғаны тану бойынша пәндік саланы талдау

Нейрондық желілер – адамның жүйке жүйесін жаңғыртуға негізделген жасанды интеллект саласындағы зерттеулердің бір бағыты. Атап айтқанда, жүйке жүйесінің қателіктерді игеру және түзету қабілеті, бұл бізге адам миының жұмысына еліктеуге мүмкіндік береді.

Биологиялық нейрондық желі жоғары байланысқа ие: бір нейронның басқа нейрондармен бірнеше мың байланысы болуы мүмкін. Бірақ бұл шамамен алынған мән, сонымен қатар әр жағдайда ол әр түрлі. Импульстарды бір нейроннан екіншісіне беру бүкіл нейрондық желінің белгілі бір қозуын тудырады. Бұл қозудың мәні нейрондық желінің кейбір кіріс сигналдарына реакциясын анықтайды. Мысалы, адамның ескі досымен кездесуі, егер кейбір жарқын және жағымды өмірлік естеліктер осы танысумен байланысты болса, нейрондық желінің күшті қозуына әкелуі мүмкін. Өз кезегінде, нейрондық желінің қатты қозуы жүрек соғу жиілігінің жоғарылауына, көздің жиі жыпылықтауына және басқа реакцияларға әкелуі мүмкін. Нейрондық желі үшін бейтаныс адаммен кездесу дерлік көрінбейді, яғни ол күшті реакциялар тудырмайды.

Нейрондық желілер адамның миы сияқты аналитикалық есептеулерді қажет ететін күрделі мәселелерді шешу үшін қолданылады. Нейрондық желілердің ең көп таралған қосымшалары:

Жіктеу жіктеу – деректерді параметрлер бойынша бөлу. Мысалы, кіруге адамдар жиынтығы беріледі және сіз кімге несие беру керектігін және кімге бермеу керектігін шешуіңіз керек. Бұл жұмысты нейрондық желі жас, төлем қабілеттілігі, несие тарихы және т.б. сияқты ақпаратты талдай отырып жасай алады.

Болжау – бұл келесі қадамды болжау мүмкіндігі. Мысалы, қор нарығындағы жағдайға негізделген акциялардың өсуі немесе құлдырауы.

Тұлғаны тану – қазіргі уақытта нейрондық желілер кеңінен қолданылады. Ол Google-де фотосуретті іздеу кезінде немесе телефонның камералары арқылы сіздің бетіңіздің жағдайын анықтап, оны ерекшелейді және тануға мүмкіндік береді.

Компьютерлік көру – басқаша айтқанда техникалық көру, объектілерді анықтауға, бақылауға және жіктеуге қабілетті машиналарды құру теориясы мен технологиясы болып табылады.

Компьютерлік көрудің мақсаты – сенсорлар арқылы алынған кескіндерді талдау негізінде нақты әлем объектілері мен көріністері туралы пайдалы тұжырымдарды қалыптастыру.

Ғылыми пән ретінде компьютерлік көру бейнелерден ақпарат алатын жасанды жүйелерді құру теориясы мен технологиясын білдіреді. Бейне деректер бейне тізбегі, әртүрлі камералардағы суреттер немесе Kinect құрылғысы немесе медициналық сканер сияқты 3D деректері сияқты көптеген нысандарда болуы мүмкін.

Технологиялық пән ретінде *Компьютерлік көру* – компьютерлік көру жүйелерін құруда компьютерлік көру теориялары мен модельдерін қолдануға тырысады.

Компьютерлік көруге деген қызығушылық жасанды интеллект саласында алғашқылардың бірі болып, теоремаларды автоматты түрде дәлелдеу және зияткерлік ойындар сияқты проблемалармен бірге пайда болды. Тіпті алғашқы жасанды нейрондық желінің архитектурасын – перцептронды – Сетчаткаға ұқсас Фрэнк Розенблатт ұсынған және оны зерттеу символдардың суреттерін тану тапсырмасының мысалында жүргізілген.

Компьютерлік көру саласындағы Прогресс екі фактормен анықталады: теорияның, әдістердің дамуы және аппараттық құралдардың дамуы. Ұзақ уақыт бойы теориялық және ғылыми зерттеулер жүйелерді практикалық қолдану мүмкіндіктерінен асып түсті.

Компьютерлік көріністің ерекшелігі – суреттерден немесе суреттер тізбегінен сипаттамаларды алу. Бұл өте пайдалы функция. Суретке түсіру процесі әдетте бұзылмайды, сонымен қатар ол өте қарапайым және қазіргі уақытта арзан. Пайдаланушылар талап ететін сипаттамалар олардың қолданылуына байланысты айтарлықтай өзгеруі мүмкін. Мысалы, қозғалыс құрылымын анықтау сияқты компьютерлік көру аспектісі суретте не бейнеленгені және камера суреттер сериясынан қалай қозғалатыны туралы түсінік алуға мүмкіндік береді. Ойын-сауық индустриясында мұндай әдістер қозғалысты экрандау және құрылымды сақтай отырып, ғимараттардың 3D-компьютерлік модельдерін құру үшін қолданылады. Бұл модельдер нақты ғимараттарды қолдануға болмайтын жерлерде қолданылады (олар өртеледі, жарылады және т.б.). Фотосуреттердің аздығымен Сіз жақсы, қарапайым, дәл және пайдаланушыға ыңғайлы модельдерді ала аласыз. Тағы бір жағдайды қарастырыңыз: мобильді роботтардың жұмысын бақылауды қалайтын адамдар. Бұл жағдайда робот қолданылатын аймақ туралы ақпарат әдетте

үлкен қызығушылық тудырмайды, тек осы аймақтағы роботтың орналасқан жері маңызды. Осылайша, дизайн туралы ақпарат жойылады және қозғалыс бақыланады, бұл роботтың нақты орнын анықтауға мүмкіндік береді.

1.2.1 Тұлғаны тану процесінің кезеңдері

Тұлғаны тану процесін бірнеше кезеңге бөлуге болады:

- фотосуретте бетті анықтау (face detection);
- бетті туралау (face alignment);
- ерекшеліктерін есептеу (features extraction);
- ерекшеліктерін салыстыру (features matching).

Тұлғаларды тану

Суреттегі бетті анықтау процессі тұлғаны танудың алғашқы қадамы болып табылады.

Бетті танудың қолданыстағы алгоритмдерін төрт санатқа бөлуге болады:

- әзірлеуші орнатқан шаблондар арқылы тану;
- эмпирикалық әдіс;
- тән инвариантты белгілер әдісі;
- сыртқы белгілер бойынша анықтау әдісі, оқыту жүйелері.

Білімге негізделген эмпирикалық тәсіл адамның бет-бейнесі деп санау үшін фотографиялық фрагментке сәйкес келуі керек ережелер жиынтығын жүзеге асыратын алгоритм құруды қамтиды. Бұл ережелер жиынтығы адамның бейнеде қалай көрінетіні және шешім қабылдау кезінде адамның не басшылыққа алатыны туралы эмпирикалық білімді рәсімдеуге тырысады.

Негізгі ережелер:

- беттің орталық бөлігі біркелкі жарықтық пен түске ие;
- беттің орталық және жоғарғы бөліктері әртүрлі жарықтылыққа ие;
- бет құрамында мұрын, ауыз және екі симметриялы орналасқан көздер бар, олар беттің қалған бөлігіне қатысты жарықтылықта күрт ерекшеленеді.

Кескінді қатты азайту әдісі шуды тегістеу және есептеу жұмыстарын азайту үшін қолданылады. Мұндай суретте жарықтылықтың біркелкі таралуы бар аймақты анықтау оңай, содан кейін жарықтылықта күрт ерекшеленетін аймақтардың ішінде болуын тексеріңіз. Дәл осы әртүрлі ықтималдық дәрежелері бар аймақтарды адам ретінде анықтауға болады.

Бет орналасқан кескін аймақтарын анықтау үшін гистограммаларды құру әдісі көлденең және тік гистограммаларды жасайды. Ал кандидат аудандарда бет әлпетін іздеу жүргізіледі. Бұл тәсіл компьютерлік көруді дамытудың басында қолданылды, өйткені ол процессордың өңдеу қуатын қажет етпеді. Жоғарыда сипатталған әдістер қарапайым біртекті фоны бар және оларды жүзеге асыру оңай кескіндегі адамдарды анықтаудың жақсы көрсеткіштеріне ие. Кейінірек көптеген ұқсас Алгоритмдер жасалды. Бірақ бұл әдістердің барлығы көптеген бет-әлпеті бар немесе күрделі фоны бар суреттерді өңдеуге мүлдем жарамсыз. Сонымен қатар, бұл әдістер бастың бұралуы мен қисаюына өте сезімтал.

Бұл әдістерде проблема екінші жағынан қарастырылады: адамның

миында болып жатқан процестерді нақты рәсімдеуге әрекет жоқ. Бұл тәсілді жақтаушылар бет бейнесінің қасиеттері мен заңдылықтарын нақты анықтауға, позицияға немесе көлбеу бұрышқа қарамастан бет әлпетінің инвариантты ерекшеліктерін табуға тырысады.

Осы әдістер тобының алгоритмдерінің негізгі кезеңдері:

- суреттерде беттің айқын белгілерін анықтау: ауыз, мұрын, көз;
- беттің шекарасын, пішінін, түсін, текстурасын, жарықтығын анықтау;
- табылған барлық белгілерді біріктіру, содан кейін оларды тексеру.

Күрделі көріністерде бет-әлпетті анықтау әдісі бет әлпетінің дұрыс геометриялық орналасуын іздеуді қамтиды. Осы мақсаттар үшін әртүрлі бағдарлар мен масштабтар бар Гаусс сүзгісі қолданылады. Содан кейін табылған белгілердің сәйкестігін және олардың өзара орналасуын кездейсоқ іздеу арқылы іздеу жүзеге асырылады.

Нысандарды топтау әдісінің мәні кескінге қызығушылық танытатын аймақтарды табу үшін Гаусс сүзгісінің екінші туындысын пайдалану болып табылады. Содан кейін шекті сүзгіні қолдана отырып, әрбір осындай аймақтың шеттері топтастырылған. Осыдан кейін анықталған белгілерді біріктіру үшін Байес желісін қолдана отырып бағалау қолданылады - осылайша бет әлпеті таңдалады.

Осы топтағы техниктер әртүрлі позицияларда тұлғаларды тану қабілетіне ие. Бірақ бетті басқа заттармен, жарықпен немесе шуылмен аздап қабаттасқан кезде тану пайызы күрт төмендейді. Күрделі кескін фоны айтарлықтай әсер етеді. Қарастырылған тәсілдер эмпиризмге негізделген. Бұл күш пен әлсіздік. Тану объектісінің жоғары өзгергіштігі, түсіру және жарықтандыру жағдайларына тәуелділігі суреттердегі бетті тануды жоғары күрделілік міндеттеріне жатқызуға мүмкіндік береді. Эмпирикалық ережелерді қолдану бет бейнесінің кейбір моделін құруға және осы тапсырманы бірнеше қарапайым тексерулерге дейін азайтуға мүмкіндік береді. Бірақ ақылға қонымды жағдайға қарамастан – сәтті жұмыс істейтін тану құралын қолдануға тырысу – адамның көру қабілеті, бірінші санаттағы әдістер тиімділігі жағынан прототиптен әлі де алыс, өйткені бұл жолды таңдаған зерттеушілер көптеген қиындықтарға тап болады. Біріншіден, үлгіні тану кезінде адамның миында болатын процестер жеткілікті зерттеліп, зерттеушілер саналы деңгейде қол жетімді адам беті туралы эмпирикалық білімнің жиынтығы мидың саналы түрде қолданатын құралдарын тауспайды. Екіншіден, адамның дәстүрлі тәжірибесі мен білімін компьютерлік ережелер жиынтығына тиімді аудару өте қиын, өйткені қатаң ережелерге сәйкес кейбір жағдайларда бет-әлпет анықталмайды және керісінше тым жалпы ережелер жалған анықтау жағдайларының көптігіне әкеледі.

Әзірлеуші орнатқан шаблондарды қолдана отырып тану (үлгіні сәйкестендіру әдістері). Үлгілер беттің белгілі бір стандартты кескінін анықтайды, мысалы, беттің жеке бөліктерінің қасиеттерін және олардың салыстырмалы жағдайын сипаттайды. Беттерді көмегімен үлгіні тексеруге әрбір суреттің сәйкестігіне берілген шаблон.

Бұл тәсілдің ерекшеліктері:

- шаблондардың екі түрі:
- деформируемый;
- қалыптаспайтын;
- алдын ала бағдарламаланған үлгілер;
- корреляция суреттегі бетті табу үшін қолданылады.

3D пішіндерін қолдана отырып, бетті тану екі аймақта жарықтылық қатынасының жұптық үлгісін қолданады. Бетті анықтау үшін берілген үлгіні салыстыру үшін бүкіл суретті беру керек. Мұны басқа масштабта жасау керек.

Анықтамалық нүктелерді тарату модельдері-бұл пішінді өзгерте алатын нысандарды білдіретін статистикалық модельдер. Әдістің пайдалы ерекшелігі-форма параметрлерінің аз санымен оқыту жиынтығынан ауыспалы нысандардың пішінін алу мүмкіндігі. Бұл дәл және ықшам параметрлеуді тиімді жіктеу жүйелерін жасау үшін пайдалануға болады.

Үлгіні танудың артықшылығы – і ске асырудың салыстырмалы қарапайымдылығы және өте күрделі емес кескіндердегі жақсы нәтижелер. Негізгі кемшілігі-үлгіні бет бейнесіне жақын калибрлеу қажеттілігі. Әр түрлі бұрыштар мен бет бұрылыстары үшін шаблондарды есептеудің үлкен күрделілігі оларды қолданудың орындылығына күмән тудырады.

Бетті сыртқы белгілері бойынша анықтау әдістері (тестілік бейнелерді өңдеу арқылы жүйені оқыту кезеңін жүргізу қажет болатын әдістер). Кескін немесе кескін фрагменті кескінді екі класқа – бетке / бетке жіктеу үшін қолданылатын белгілі бір есептелген белгілер векторымен байланысты. Әдетте, бет кескінінің математикалық моделін құруға негізделген әдістерді қолдана отырып, кескіннен бетті іздеу барлық өлшемдегі кескіннің барлық тікбұрышты фрагменттерін сұрыптаудан және әр фрагменттің бет-әлпетін тексеруден тұрады. Бірақ толық іздеу схемасында артықтық және жоғары есептеу күрделілігі сияқты кемшіліктер болғандықтан, авторлар қарастырылатын бөліктердің санын азайту үшін әртүрлі әдістерді қолданады.

Әдістеменің негізгі принциптері:

Схоластика. Яғни, әр кескін терезе арқылы сканерленеді және кейбір мәндер векторларымен ұсынылады.

Блок құрылымы. Кескін әртүрлі масштабтағы қиылысатын немесе қиылыспайтын бөлімдерге бөлінеді және бағалау векторлардың салмақтарын бағалау алгоритмдерін қолдану арқылы жүзеге асырылады.

Алгоритмдерді оқыту үшін беті мен беті жоқ қолмен дайындалған суреттер жиынтығы қажет.

Сондай-ақ, ең маңызды міндет – күшті жіктеуіштерді таңдау, себебі олар суретте анықталған белгілерді тексеру үшін ең жоғары басымдыққа ие болады. Әлсіз жіктеуіштердің санын олардың бір-біріне ұқсастығына байланысты азайту керек, сонымен қатар шу шығарындыларынан туындаған жіктеуіштерді жою жоғарыда талқыланды. Хаар функциялары сәйкесінше X және Y осьтері бойымен жарықтылық айырмашылығының нүктелік мәнін береді.

Белгілері бар терезені сканерлеу алгоритмі келесідей. Зерттелетін сурет, сканерлеу терезесі бар және пайдаланылған функциялар таңдалады. Содан кейін сканерлеу терезесі терезенің 1 ұяшығына қадам жасай отырып, кескін бойынша дәйекті түрде жылжи бастайды. Сканерлеу әр түрлі масштабтар үшін дәйекті түрде жасалады және кескіннің өзі емес, сканерлеу терезесі масштабталады. Табылған барлық белгілер адамның табылғанын немесе табылмағанын шешетін классификаторға түседі.

Төмен қуатты дербес компьютерлердегі барлық белгілерді есептеу мүмкін емес. Сондықтан классификатор тек қажетті белгілер жиынтығына жауап беруі керек. Сондықтан жіктеуішті белгілі бір ішкі жиынның қырларын табуға үйрету керек. Мұны компьютерлік машинаны автоматты түрде оқыту арқылы жасауға болады.

Алгоритм контекстінде сыныптарға бөлінген көптеген суреттер бар. Олардың қай сыныпқа жататыны белгілі бірнеше суреттер бар, мысалы, "мұрынның фронтальды позициясы"класы. Бұл бірнеше суреттер Оқу үлгісі болып табылады. Қалған суреттердің класы белгісіз. Бастапқы жиынтықтан еркін нысанды жіктеуге қабілетті алгоритм құру қажет. Бұл мәселені шешу үшін өнімділікті арттыру технологиясы бар. Күшейту-аналитикалық модельдердің дәлдігін арттыратын әдістер жиынтығы. Жіктеу қателіктерінің аз мөлшерін беретін Модель "күшті"деп аталады. "Әлсіз" сенімді жіктеуге мүмкіндік бермейді және жұмыста көптеген қателіктер жібереді.

Тұлға бойынша іздеуді арттыру алгоритмі:

- Тік бұрышты белгілер негізінде әлсіз жіктеуіштерді анықтау;
- Сканерлеу терезесінің әр қозғалысы үшін әр мысал үшін тікбұрышты функция есептеледі;
- Әр функция үшін ең қолайлы шекті таңдаңыз;
- Ең жақсы мүмкіндіктер мен ең жақсы шегі таңдалады;
- Үлгі тым салмақты.

Каскадтық моделі қатаң сыныптауышын білдіреді ағаш шешімдер, онда әрбір торабы салынды болатындай анықтайтын қызықтыратын барлық үлгілер және отбрасывать облысы болып табылмайтын шаблондарды. Ағаш түйіндері түйін ағаштың тамырына неғұрлым жақын болса, соғұрлым примитивтер аз болады, сондықтан шешім қабылдауға аз уақыт кетеді. Каскадты модельдің бұл түрі суреттерді өңдеуге өте ыңғайлы, онда табылған суреттердің саны аз. Бұл жағдайда әдіс берілген аймақта кескін жоқ екенін тез шешіп, келесіге өтуі мүмкін.

Егер жүйенің кірісіне түрлі-түсті кескін берілсе, онда егер кескін түсті кодтауды қолдана отырып алдын-ала өңделген болса, алгоритмнің жылдамдығын едәуір арттыруға болады. Түсті кодтау жалған позитивтердің санын азайтуға көмектеседі.

Бүгінгі таңда Виола-Джонс алгоритмі жоғары дәлдігімен және жоғары жылдамдығымен танымал.

Осы тапсырмаларды орындаудың негізгі әдістері:

- Жасанды нейрондық желілер;

- Негізгі компоненттер әдісі;
- Факторлық талдау әдісі;
- Сызықтық дискриминантты талдау;
- Тірек векторлар әдісі;
- Аңғалдық Байес жіктеуіші;
- Марковтың жасырын модельдері;
- Тарату әдісі;
- Сирек терезе желісі;
- Белсенді модельдер;
- Виола-Джонс және т. б.

Олардың кейбіреулерінің ерекшеліктерін қарастырайық. Бүгінгі таңда жасанды нейрондық желілер әдісі-бетті тану мәселелерін шешудің ең көп таралған әдісі. жасанды нейрондық желі-бұл өзара байланысты және өзара әрекеттесетін нейрондар жүйесі болып табылатын математикалық модель. Нейрондық желілер сөздің әдеттегі мағынасында бағдарламаланбайды, олар оқытылады. Техникалық тұрғыдан оқыту нейрондар арасындағы байланыс коэффициенттерін (синапстарды) табудан тұрады.

Тірек векторлық Машина объектілердің оқу жиынтығының ақпараттылығын айтарлықтай жоғалтпай, белгілер кеңістігінің өлшемін азайту үшін қолданылады. Сызықтық кеңістіктегі векторлар жиынтығына негізгі компоненттер әдісін қолдану кеңістіктің негізіне өтуге мүмкіндік береді, осылайша жиынтықтың дисперсиясы негізгі осьтер деп аталатын алғашқы бірнеше осьтер бойымен бағытталады. Осылайша алынған негізгі осьтерге созылған ішкі кеңістік барлық кеңістіктер арасында оңтайлы болып табылады, өйткені ол оқыту жиынтығын жақсы сипаттайды. Бұл жіктеу және регрессиялық талдау үшін қолданылатын бақыланатын оқыту алгоритмдеріне ұқсас Алгоритмдер жиынтығы. Бұл әдіс сызықтық классификаторлар тобына жатады. Қолдау векторлық машинасы сызықтық сынып бөлінуіне негізделген.

Бүгінгі таңда Виола-Джонс әдісі жоғары өнімділік, жалған дабылдардың төмен жиілігі және бет-әлпетті дұрыс танудың жоғары пайызы тұрғысынан ең перспективалы болып табылады.

Әдіс негізделген негізгі принциптер:

- суреттер интегралды көріністе қолданылады, бұл бетті тез табуға мүмкіндік береді;
- Хаар белгілері қолданылады, осы белгілердің көмегімен бет пен оның ерекшеліктерін іздеу жүзеге асырылады;
- Күшейту суреттегі қалаған объект үшін ең қолайлы функцияларды таңдау үшін қолданылады;
- жауап беретін классификатор қолданылады: адам / адам емес;
- Беті жоқ терезелерді тез жою үшін Хаара белгілерінің каскадтарын қолданады.

Жіктеуіштер өте баяу үйренеді, бірақ беттер өте тез анықталады. Егер қалаған объект суретте шамамен 30 градусқа дейін сәл бұрышта болса, Алгоритм жақсы жұмыс істейді. Жоғары көлбеу бұрыштарда анықтау

жылдамдығы әлдеқайда төмен. Өкінішке орай, бұл стандартты іске асыруда адамның бет-әлпетін 30 градустан жоғары бұрышпен анықтауға мүмкіндік бермейді, бұл қазіргі алгоритмді қазіргі бет-әлпетті тану жүйелерінде қолдануды қиындатады немесе мүмкін болмайды.

Сандық суретте адамның бетін анықтау міндеті келесідей. Беті бар сурет бар. Ол екі өлшемді пиксель матрицасымен ұсынылған, онда әр пиксель 0-ден 255-ке дейін, егер кескін қара және ақ болса, егер кескін түрлі-түсті болса 0-ден 2553-ке дейін болады. Алгоритм тұлғаларды анықтап, оларды белгілеуі керек - іздеу тікбұрышты белгілері бар кескіннің белсенді аймағында жүзеге асырылады, оның көмегімен табылған тұлға сипатталады. Қарапайым сөзбен айтқанда, сканерлеу терезесі қолданылады. Кескін іздеу терезесінде сканерленеді, содан кейін классификатор әр позицияға қолданылады.

Деректермен кез-келген әрекетті орындау үшін Виола-Джонс әдісінде суреттердің интегралды көрінісі қолданылады. Интегралдық көрініс суреттегі еркін тікбұрышты аймақтың жалпы жарықтығын тез есептеуге мүмкіндік береді және бұл аймақтың көлеміне қарамастан, есептеу уақыты тұрақты болып қалады.

Кескіннің интегралдық көрінісі-бұл бастапқы кескінмен бірдей мөлшердегі матрица. Оның элементтерінің әрқайсысы осы элементтің үстінде және сол жағында орналасқан барлық пикселдердің жарықтығын сақтайды.

1.2.2 Бетті түзету функциясы

Бет кескіннен табылғаннан кейін (жоғарғы сол жақ бұрыштың координаттары, тіктөртбұрыштың ұзындығы мен ені алынады), суреттерді алдын-ала өңдеу керек. Шуды азайту үшін әртүрлі сүзгілерді қолдануға болады (медиан, Гаусс және т.б.). Сонымен қатар, суретті қалыпқа келтіру процедурасын орындау керек, яғни.көздің орталықтарын байланыстыратын сызықты кесіп, масштабтаңыз және көлденең күйге бұрамыз. Бұл жұмыста алдын-ала өңдеуге арналған теориялық материалға аз көңіл бөлінгенімен, жүйенің сапасы кіріс кескіндерін алдын-ала өңдеуге байланысты екенін атап өткен жөн. Жұмыс барысында сүзу кезеңінің болмауы дұрыс тексеру ықтималдығының төмендеуіне алып келеді.

1.2.3 Сипаттамаларды есептеу және салыстыру

Соңғы кезең-белгілерді есептеу және олардың коллекцияларын бір-бірімен салыстыру. Төменде ең танымал алгоритмдердің сипаттамасы келтірілген.

Серпімді графиктерді салыстыру.

Әдістің мәні-бет суреттерін сипаттайтын серпімді графиктерді салыстыру. Беттері өлшенген жиектері мен шыңдары бар графиктер түрінде ұсынылған. Тану кезеңінде сілтеме графигі өзгеріссіз қалады, ал екіншісі сілтеме графигіне жақсы сәйкес келеді. Осы әдіске негізделген тану

жүйелерінде графиктер тікбұрышты тор немесе беттің антропометриялық нүктелерінен құралған құрылым болуы мүмкін.

Графиктің шыңдары әдетте Габор сүзгілерінің күрделі мәндерін қолдана отырып немесе Габор толқындарын қолдана отырып есептеледі, олар графиктің шыңының белгілі бір жергілікті аймағында Габор сүзгілерін қолдана отырып пиксель жарықтығының мәндерін жинақтау арқылы есептеледі.

Графтың шеттері көршілес шыңдар арасындағы қашықтық бойынша өлшенеді. Екі бағанның арасындағы қашықтық бағаның деформациясының белгілі бір функциясын қолдана отырып есептеледі, ол шыңдарда есептелген белгілердің мәндері мен графиктің жиектерінің деформация дәрежесі арасындағы айырмашылықты ескереді.

График оның шыңдарының әрқайсысын бастапқы орнына қатысты белгілі бір бағытта белгілі бір қашықтыққа ығыстыру және шыңның орнын таңдау арқылы деформацияланған графиктің шыңдарындағы белгілер мен тірек графигінің тиісті шыңдарының арасындағы айырмашылық ең аз болады. Бұл операция анықтамалық және деформацияланатын графиктердің сипаттамалары арасындағы ең аз жалпы айырмашылыққа жеткенге дейін графиктің әр шыңы үшін жасалады. Деформацияланатын графиктің осы позициясындағы деформация бағасының функциясының мәні сілтеме графигі мен бет кірісі арасындағы айырмашылықтың өлшемі болып табылады. Бұл деформация процедурасы жүйелік дерекқорда қол жетімді барлық тірек беттері үшін орындалуы керек. Осы тану жүйесінің жұмысының нәтижесінде біз баға деформациясы функциясының ең жақсы мәні бар стандартты аламыз.

Кейбір дереккөздерде фотосуреттерде және бет әлпетінде 15 градусқа дейін әр түрлі өрнектерде де танудың 95-97% тиімділігін көрсетеді. Алайда, мұндай жүйелерді жасаушылар бұл жүйе үлкен есептеу қуатын қажет етеді дейді.

2. Бағдарламалық жасақтаманы таңдау негіздемесі

Бағдарламалық жасақтаманы таңдауды негіздеудің мақсаты-мәселені шешудің заманауи тәсілдері және жағдайды ең жақсы және ыңғайлы шешу үшін бағдарламалық жасақтаманың сипаттамасы. Көптеген жаңа технологиялар мен бағдарламалық жасақтама қосымшалары жасауға мүмкіндік береді, бірақ бұл әртүрлілік олардың біркелкі көрінуін қиындатады, ал белгілі бір бағдарламалық жасақтаманың жұмыс сапасындағы қажетті технологиялармен айырмашылық жұмысқа қосымша шектеулер қояды.

Даму ортасымен жұмыс істеудің ыңғайлылығы әзірлеушінің бағдарламалық жасақтамадағы жұмысының сапасын арттыруға және жылдамдығын арттыруға мүмкіндік береді, сондықтан IDE-нің барлық артықшылықтары мен кемшіліктерін егжей-тегжейлі қарастыру қажет.

2.1 Бағдарламалау тілін таңдау

2.1.1 C ++ бағдарламалау тілі

C ++ басталуы 1979 жылдан басталады, сол кезде Bell AT&T-де жұмыс істеген Бьярне Страуструп с тілінде сыныптармен жұмыс істей бастады. Ол simula, Ada, ML, CLU және ALGOL сияқты көптеген басқа тілдерден функцияларды алады.

Сонымен, C тілінің ерекшелігімен қатар, с ++ сыныптарды, күрделі типті тексеруді, әдепкі функцияның дәлелін және негізгі мұрагерлікті де қамтыды. 1983 жылға дейін ол C сыныптарымен, ал 1983 жылы с ++деп аталды. 1998 жылы ANSI-ISO бірлескен комитеті C ++тіліне арналған спецификацияны шығарды.

2011 жылдың ортасында C ++ 11 тілінде жаңа стандарт шығарылды. Бұл Boost кітапханасының дизайнына айтарлықтай әсер етті және көптеген жаңа модульдер тиісті Boost кітапханаларынан тікелей алынды. Стандарт сонымен қатар толық рандомизация кітапханасын, тұрақты өрнектерді қолдауды, жаңа с++ синхрондау кітапханасын, стандартты ағындар кітапханасын, автоматтандыруды қолдауды, auto кілт сөзін, жақсартылған қосылысты қолдауды және массивтерді баптандыру тізімдерін қоса алғанда, жаңа мүмкіндіктерді қосты. контейнерлердің жаңа үлгілері мен сыныптары. 2014 жылдың желтоқсанында шығарылған с ++ 14 C ++ 11-ге қарағанда кішігірім жақсартулар мен қателерді түзетуді қамтыды.

C ++ бастапқыда арнайы конструкцияларды бастапқы кодқа түрлендіретін с препроцессорына ұқсас "алдын-ала жасаушы" болды.

Pre-c++ компиляторы оқыған" алдын-ала код" әдетте .cc файлында болды . C немесе .cpp. Содан кейін бұл файл кеңейтімі бар Бастапқы с файлына айналады .c.

Бастапқы с ++ файлдарының номенклатурасы өзгеріссіз қалады -. сс кеңейтімдері әлі де қолданылады және .cpp. Алайда, нақты уақыттағы с ++ прекурсорының жұмысы әдетте компиляция процесінің өзі кезінде жасалады. Көбінесе компиляторлар бастапқы файлда қолданылатын тілді оны кеңейту арқылы анықтайды. Бұл бастапқы с++ кеңейтімін қабылдайтын Borland және Microsoft с++компиляторларына қатысты . cpp.

С ++ бағдарламасының бағдарламалау туралы көптеген артықшылықтары бар. Барлық офлайн с ++ бағдарламалық файлдары бағдарламаға қосымша шешімдерді іске қосуға мүмкіндік беретін негізгі функцияны қолдануы керек.

С ++ бағдарламасы автономды және біріктірілген файлдардың тіркесімі болып табылатын бірлестіктер мен құрылымдарды қолдай алады. Ғаламдық айнымалылар мен ғаламдық функциялар С ++ тілінде қолданылады, бірақ олар көптеген басқа жоғары деңгейлі тілдерде қолданылмайды және бұл бағдарламалау тілі үшін үлкен артықшылық болып табылады.

С ++ нысандарды пайдаланбайтындықтан, басқа бағдарламада бөлінген жадта сақталған ақпаратты өзгерту мүмкіндігі бар бағдарламаны құру қиынға соғады.

С ++ бағдарламалары көп парадигмалық бағдарламалауды қолданады және үш парадигманы ұстанады: ООР (объектіге бағытталған бағдарламалау), ор (әмбебап бағдарламалау), IP (императивті бағдарламалау).

Нысанға бағытталған (және объектіге бағытталған) бағдарламалау с++бағдарламаларында жиі қолданылатын стратегияға қарағанда бағдарламалау міндеттеріне сәл өзгеше көзқарасты қолданады. С ++ бағдарламалау тапсырмалары әдетте "процедуралық тәсілді" қолдана отырып шешіледі: тапсырма ішкі тапсырмаларға бөлінеді және бұл процесс ішкі тапсырмалар бағдарламаланғанға дейін қайталанады. Бұл аргументтер арқылы ғаламдық немесе жергілікті (статикалық) айнымалыларды беру арқылы жұмыс істейтін функциялар жиынтығын жасайды.

Нысанға бағытталған тәсіл сынып атауында қолданылатын кілт сөздер арқылы жүзеге асырылады. Содан кейін бұл кілт сөздер Ішкі иерархияны бейнелеу үшін әртүрлі диаграммаларда бейнеленген. Кілт сөздер объектілерге айналады, ал иерархия сол объектілер арасындағы қатынасты анықтайды. Мұнда "объект" термині объект туралы барлық ақпаратты қамтитын шектеулі, анықталған құрылымды сипаттау үшін қолданылады: деректер түрлері және деректерді басқару функциялары. Нысанға бағытталған тәсілдің мысалы:

Автодилерлер мен автогаранттардың жұмысшылары мен иелеріне келесі сомалар төленеді. Біріншіден, гаражда жұмыс істейтін механиктерге ай сайын белгілі бір сома төленеді. Екіншіден, компанияның иесі ай сайын белгіленген соманы алады. Үшіншіден, автосалонда жұмыс істейтін және ай сайынғы жалақы мен сатылған автомобиль үшін сыйақы алатын автомобиль

сатушылары бар. Ақырында, компания саяхаттайтын пайдаланылған автокөлік сатып алушыларды пайдаланады; бұл қызметкерлер ай сайынғы жалақы, сатып алынған автомобиль үшін сыйлықақы алады және іссапар шығындарын өтейді.

Жоғарыда аталған жалақыны басқаруды ұсынған кезде, кілт сөздер механика, иесі, сатушылар және сатып алушылар болуы мүмкін. Мұндай бірліктердің қасиеттері: ай сайынғы жалақы, кейде сатып алу немесе сату үшін сыйлықақы, кейде жол шығындарын өтеу. Мәселені осылайша талдай отырып, біз келесі көзқарасқа келеміз:

Автодилерлер мен автогаранттардың жұмысшылары мен иелеріне келесі сомалар төленеді. Біріншіден, гаражда жұмыс істейтін механиктерге ай сайын белгілі бір сома төленеді. Екіншіден, компанияның иесі ай сайын белгіленген соманы алады. Үшіншіден, автосалонда жұмыс істейтін және ай сайынғы жалақы мен сатылған автомобиль үшін сыйақы алатын автомобиль сатушылары бар. Ақырында, компания саяхаттайтын пайдаланылған автокөлік сатып алушыларды пайдаланады; бұл қызметкерлер ай сайынғы жалақы, сатып алынған автомобиль үшін сыйлықақы алады және іссапар шығындарын өтейді.

Жоғарыда аталған жалақыны басқаруды ұсынған кезде, кілт сөздер механика, иесі, сатушылар және сатып алушылар болуы мүмкін. Мұндай бірліктердің қасиеттері: ай сайынғы жалақы, кейде сатып алу немесе сату үшін сыйлықақы, кейде жол шығындарын өтеу. Мәселені осылайша талдай отырып, біз келесі көзқарасқа келеміз:

Иесі мен механигі айына белгіленген жалақы алатын бір түрі болуы мүмкін. Осы типтегі тиісті ақпарат ай сайынғы сома болады. Сонымен қатар, бұл нысанда аты, мекен-жайы және әлеуметтік сақтандыру нөмірі сияқты мәліметтер болуы мүмкін.

Автосалонда жұмыс істейтін автомобиль сатушылары жоғарыдағыдай ұсынылуы мүмкін, бірақ кейбір қосымша функциялары бар: мәмілелер саны (сату) және мәміле бонусы.

-Нысандар иерархиясында біз автодилерлерге иесі мен механиктен "алуға" мүмкіндік беретін алғашқы екі объектінің арасындағы қатынасты анықтаймыз. Соңында, пайдаланылған автомобильдерді сатып алушылар бар. Олар іссапар шығындарын қоспағанда, сатушының функционалдығын бөліседі. Осылайша, Қосымша функционалдылық қымбатқа түседі және бұл түрі сатушылардан алынады.

Бұл тілдің басты кемшілігі-оның күрделілігі мен бағдарламаны құру ерекшелігі, сонымен қатар бағдарламаға өңделген деректерді жинау проблемалары, бұл көбінесе әртүрлі сыни қателіктерге әкеледі.

2.1.2 Python бағдарламалау тілі

Python – бұл жоғары деңгейлі, интерпретацияланған және әмбебап динамикалық бағдарламалау тілі, ол кодты оқуға бағытталған. Python

синтаксисі бағдарламашыларға Java немесе C ++кодтарына қарағанда азырақ кодты бағдарламалауға көмектеседі. 1991 жылы Guido van Rossum әзірлеушісі негізін қалаған тіл бағдарламалаудың қарапайым және қызықты тәсіліне ие. Python көптеген бағдарламалау парадигмаларына байланысты ірі ұйымдарда кеңінен қолданылады. Олар әдетте әмбебап, объектіге бағытталған, функционалды бағдарламалауды қамтиды. Онда жадты автоматты басқару және динамикалық функциялары бар кең және үлкен стандартты кітапхана бар.

Python көптеген ғылыми зерттеулерге арналған тілдік франка болды. Ол әмбебап бағдарламалау тілдерінің артықшылықтарын Matlab немесе R. Python сияқты бағдарламаланатын тілдерді қолданудың қарапайымдылығымен үйлестіреді, деректерді жүктеу, визуализация, Статистика, өңдеу, суретті өңдеу және тағы басқалар үшін кітапханалар бар. Бұл жан-жақты құралдар жиынтығы әзірлеушілерге жалпы және арнайы мақсаттағы функциялардың үлкен жиынтығын ұсынады. Жалпы мақсаттағы бағдарламалау тілі ретінде Python сонымен қатар күрделі графикалық пайдаланушы интерфейстерін (GUI), веб-қызметтерді құруға және қолданыстағы жүйелерге интеграциялауға мүмкіндік береді.

Python аудармашысы және кең стандартты кітапхана Python веб-сайтындағы барлық негізгі платформалар үшін бастапқы немесе екілік түрінде еркін қол жетімді, https://www.python.org/, және қоғамдастықтың барлық мүшелері арасында еркін таратылады.

Python аудармашысы C немесе C ++ (немесе C-мен байланысты басқа тілдерде) енгізілген жаңа мүмкіндіктер мен деректер түрлерімен оңай кеңейеді. Python сонымен қатар қолданушы қосымшалары үшін кеңейту тілі ретінде жарамды.

Python қолдану оңай, бұл нақты бағдарламалау тілі, дегенмен кейбіреулер оны сценарий тілі деп атайды, ол қабық сценарийлері немесе пакеттік файлдар ұсынғаннан гөрі үлкен бағдарламалар үшін әлдеқайда көп құрылым мен қолдауды ұсынады. Екінші жағынан, Python сонымен қатар C++ - ке қарағанда қателерді дәлірек тексеруді ұсынады және өте жоғары деңгейлі тіл бола отырып, икемді массивтер мен сөздіктер сияқты жоғары деңгейлі мәліметтер типтері бар. Деректердің жалпы түрлеріне байланысты Python Awk немесе тіпті Perl-ге қарағанда әлдеқайда проблемалық аймаққа қолданылады, бірақ Python-да көптеген тілдер сияқты оңай дамиды.

Python сіздің бағдарламаңызды басқа Python бағдарламаларында қайта пайдалануға болатын модульдерге бөлуге мүмкіндік береді. Ол стандартты модульдердің үлкен жиынтығымен бірге келеді, оларды сіз өзіңіздің бағдарламаларыңыздың негізі ретінде немесе Python бағдарламалауын үйренуді бастау үшін мысалдар ретінде пайдалана аласыз. Бұл модульдердің кейбіреулері файлды енгізу-шығару, жүйелік қоңыраулар, розеткалар және тіпті Tk сияқты графикалық пайдаланушы интерфейстерінің жиынтығы үшін интерфейстер сияқты функцияларды ұсынады. Python-бұл интерпретацияланған тіл, ол сізге бағдарламаны жасау кезінде көп уақытты

үнемдей алады, өйткені компиляция қажет емес. Аудармашыны интерактивті түрде қолдануға болады, бұл тілдік функциялармен тәжірибе жасауды, бағдарлама эскиздерін жазуды немесе бағдарламаны төменнен жоғары қарай жасау кезінде функцияларды тексеруді жеңілдетеді.

Python бағдарламаларды ықшам түрде жазуға мүмкіндік береді. Python бағдарламалары, әдетте, бірнеше себептерге байланысты C ++ немесе Java бағдарламаларына қарағанда әлдеқайда қысқа:

- Мәліметтер типтері күрделі операцияларды бір өрнекте білдіруге мүмкіндік береді.
- Операторларды топтау жетекші және соңғы жақшалардың орнына шегініс арқылы жүзеге асырылады;
- Айнымалыларды немесе аргументтерді жариялау қажет емес.

Python кеңейтілетін болып табылады: Егер сіз C-де қалай бағдарламалауды білсеңіз, максималды жылдамдықта сыни операцияларды орындау үшін аудармашыға жаңа кіріктірілген функцияны немесе модульді қосу немесе Python бағдарламаларын тек екілік түрінде қол жеткізуге болатын кітапханалармен байланыстыру оңай (мысалы, белгілі бір жеткізушінің графикалық кітапханасы).

Егер сіз шынымен қосылған болсаңыз, Сіз Python аудармашысын C++ тілінде жазылған қосымшамен байланыстыра аласыз және оны осы бағдарлама үшін кеңейтім немесе командалық тіл ретінде қолдана аласыз.

Python пакеттерінің индексі, қысқаша PyPI, сонымен қатар Cheese Shop деп те аталады, Python бағдарламалық жасақтамасының ресми үшінші тарап репозиторийі болып табылады. Бұл Cpan, Perl репозиторийіне ұқсас. Бірнеше пакет менеджерлері, соның ішінде pip, PyPI-ді пакеттер мен олардың тәуелділіктері үшін әдепкі көз ретінде пайдаланады. 113000-нан астам Python пакеттері PyPI арқылы қол жетімді.

PyPI негізінен Python пакеттерін sdists (бастапқы таратулар) деп аталатын мұрағат ретінде орналастырады. PyPI пайдаланушыларға ақысыз бағдарламалық жасақтама лицензиясы немесе POSIX үйлесімділігі сияқты метадеректер негізінде кілт сөздер немесе сүзгілер бойынша пакеттерді іздеуге мүмкіндік береді. Бір PyPI жазбасы бүкіл пакеттен және оның метадеректерінен басқа, пакеттің алдыңғы нұсқаларын, алдын-ала құрастырылған кітапханаларды (мысалы, Windows-та DLL кітапханалары бар) және әртүрлі операциялық жүйелер мен Python нұсқаларына арналған әртүрлі графикалық формаларды сақтай алады.

Python бағдарламалық жасақтама компанияларында ойындар, веб-шеңберлер, тілдерді дамыту, прототиптеу, графикалық қосымшалар және басқалар сияқты көптеген қосымшаларға ие. Оның кейбір артықшылықтары:

Кең қолдау кітапханалары. Тіл стандартты кітапханаларды ұсынады, оның ішінде жол операциялары, Интернет, веб-қызмет құралдары, интерфейстер және операциялық жүйенің хаттамалары. Күнделікті бағдарламалау тапсырмаларының көпшілігі Python кодының ұзындығын қысқартатын етіп енгізілген.

Интеграция. Python COM немесе COBRA компоненттерін шақыру арқылы веб-қызметтерді дамытуды жеңілдететін корпоративті қосымшаларды интеграциялауды қолданады. Ол қуатты басқару қабілетіне ие, өйткені ол командаларды тікелей C, C ++ немесе Java арқылы Jython арқылы шақырады. Python сонымен қатар XML және басқа белгілеу тілдерін өңдейді, өйткені ол барлық заманауи операциялық жүйелерде бірдей байт кодымен жұмыс істей алады.

Өнімділік. Технологиялық интеграцияның қуатты функциялары, блокты тестілеу және басқарудың жетілдірілген мүмкіндіктері арқасында көптеген қосымшалар тезірек және нәтижелі жұмыс істей алады. Бұл масштабталатын көп протоколды желілік қосымшаларды құрудың тамаша нұсқасы.

Бұл бағдарламалау тілінің кемшілігі-оны жүзеге асыру, яғни компилятордың орнына аудармашыны қолданады, нәтижесінде ол мобильді платформаларда қолданылмайтын жұмыстың баяулауына әкеледі. Сонымен қатар, Python тілі динамикалық түрде терілген, сондықтан оның көптеген шектеулері бар. Бұл бағдарламаны тестілеу уақытын арттырады, өйткені қателерді бақылау мүмкіндігі бағдарлама іске қосылғаннан кейін ғана болады.

2.1.3 Java бағдарламалау тілі

Java-ның басталуы 1991 жылдан басталады, Патрик Нотон мен Джеймс Гослинг бастаған Sun инженерлер тобы кабельдік теледидар қосқыштары сияқты тұтынушылық құрылғылар үшін қолдануға болатын шағын компьютерлік тілді жасағысы келді. Бұл құрылғылар үлкен өнімділікке және үлкен жадқа ие болмағандықтан, тіл аз болуы керек және өте қатаң код жасайды. Сонымен қатар, әртүрлі өндірушілер әртүрлі орталық процессорларды (процессорларды) таңдай алатындықтан, тілдің кез-келген архитектураға байланбауы маңызды. Жоба "Жасыл" код атауымен ұйымдастырылды. Шағын, қатаң және кросс-платформалық кодқа қойылатын талап командаңы есептеудің алғашқы күндерінде Паскальда кейбір жүзеге асырылған модельді еске түсіруге итермеледі. Никлаус Вирт, Паскальдың өнертапқышы, ықтимал машина үшін аралық кодты құратын портативті тілді дамытуда ізашар болды. Мұндай машиналарды көбінесе виртуалды машиналар деп атайды, демек Java немесе JVM виртуалды машинасы. Аралық кодты кез-келген машинада дұрыс аудармашымен қолдануға болады. Жасыл инженерлер өздерінің негізгі мәселесін шешетін виртуалды машинаны да қолданды. Sun әзірлеушілері Паскальдан гөрі с ++ тілін негізгі бағдарламалау тілі ретінде таңдады. Атап айтқанда, олар тілді процедуралық емес, объектіге бағытталған етіп жасады. Гослинг өз тілін "емен" деп атауға шешім қабылдады (мүмкін, ол күн сәулесіндегі ғимараттың терезесінен тыс емен түрін ұнатқан шығар). Кейінірек сол

атаумен бағдарламалау тілі бар екендігі белгілі болды және ол Java деп аталды.

Тілді ертерек енгізу үшін Hotjava шолушысы Java-ның артықшылықтарын көрсету үшін жасалды. Әзірлеушілер сонымен қатар апплеттердің қалай жұмыс істейтінін көрсетті. Бұл "технологияның дәлелі" 1995 жылы 23 мамырда SunWorld '95-те көрсетілді және презентациядан кейін Java тіліне деген қызығушылықтың өсуі байқалды, ол бүгінгі күнге дейін жалғасуда.

1996 жылы Java-ның алғашқы шығарылымы тек компьютерлік баспасөзде ғана емес, сонымен қатар The New York Times, The Washington Post және Business Week сияқты әлемдік БАҚ-та керемет толқулар тудырды. Java-бұл бірінші және жалғыз бағдарламалау тілі.

Java синтаксисі - бұл C ++синтаксисінің шынымен таза нұсқасы. Тақырып файлдары, көрсеткіш арифметикасы (немесе тіпті көрсеткіш синтаксисі), құрылымдар, бірлестіктер, операторлардың шамадан тыс жүктелуі, виртуалды базалық сыныптар және т.б. қажет емес. Мысалы, switch операторының синтаксисі Java-да өзгермейді. Бұл C ++ тілін білетін бағдарламашылар үшін ауысуды жеңілдетеді, сіз Java синтаксисіне оңай ауыса аласыз.

Егер сіз визуалды бағдарламалау ортасына үйреншікті болсаңыз (мысалы, Visual Basic), Java бұл жағынан оңай болмайды. Visual Basic-тің артықшылығы-визуалды даму ортасы қосымшаның инфрақұрылымының көп бөлігін автоматты түрде қамтамасыз етеді. Ұқсас функцияны Java-да қолмен кодтау керек, сондықтан визуалды даму жеткіліксіз дамыған. Алайда, үшінші тараптың даму орталары апарып тастау сияқты тілмен өзара әрекеттесуді жеңілдететін функцияларды қолдайды.

Тілдің тағы бір аспектісі-бұл ықшам және платформалық. Java мақсаттарының бірі-шағын машиналарда дербес жұмыс істей алатын бағдарламалық жасақтама жасау. Аудармашының негізгі өлшемі және сыныпты қолдау шамамен 40 мың. Негізгі Стандартты кітапханалар мен ағындарды қолдау (автономды микроядро) 175 КБ құрайды. Ол кезде бұл үлкен жетістік болды. Содан бері кітапхана үлкен мөлшерге жетті. Қазіргі уақытта жеке Java Micro Edition басылымы бар, ендірілген құрылғыларға жарамды шағын кітапханасы бар.

Java Артықшылықтары:

- Оңай үйрену. Java басқа бағдарламалау тілдеріне қарағанда жазуды, құрастыруды, күйін келтіруді және үйренуді жеңілдету үшін жасалған.

Ол Кросс-платформалы. Java-ның маңызды артықшылықтарының бірі-оның бір компьютерлік жүйеден екіншісіне оңай көшу мүмкіндігі. Әр түрлі жүйелерде бірдей бағдарламаны іске қосу мүмкіндігі Бүкіләлемдік Интернет бағдарламалық жасақтамасы үшін өте маңызды, ал Java платформадан тәуелсіз бола отырып, оны сәтті етеді.

- Қауіпсіздік. Java тілдің бөлігі ретінде қауіпсіздікті қамтиды. Java

тілі, компилятор, аудармашы және жұмыс уақыты қауіпсіздікті ескере отырып жасалған.

- Сенімділігі. Java қателерді ертерек тексеруге көп көңіл бөледі, өйткені Java компиляторлары жұмыс уақытында басқа тілдерде пайда болатын көптеген мәселелерді анықтай алады.

Көп ағынды – бұл бағдарламаның бір бағдарламада бір уақытта бірнеше тапсырманы орындау мүмкіндігі. Java-да көп ағынды біріктіру оңай болды, ал басқа тілдерде көп ағынды қосу үшін операциялық жүйеге тән процедураларды шақыру қажет.

Осы артықшылықтар мен тілдің осы тезисті жүзеге асыру үшін қажетті бағдарламалық жасақтамамен үйлесімділігінің арқасында Java тілі негізгі бағдарламалау тілі ретінде таңдалды.

2.2 Бағдарламалық жасақтаманы жасау ортасын таңдау

Бағдарламалық жасақтаманы әзірлеуде жасау ортасы үлкен рөл атқарады. Оның көмегімен әзірлеуші бағдарламашы кодпен, жоба құрылымымен өзара әрекеттеседі, жасалған бағдарламалық жасақтаманы жөндейді және сынайды.

Жасау ортасын таңдағанда келесі факторларды ескеріңіз:

- Жобада қолданылатын технологияларды дамыту ортасын қолдау;
- Интерфейс ыңғайлылығы;
- Кодты өңдеуге арналған кіріктірілген құралдардың болуы (синтаксисті бөлектеу, кодты автоматты түрде аяқтау, кодты пішімдеу);
- Қолданбалы процестерді диагностикалау, бақылау құралдарының болуы (жад объектілерін пайдалануды бақылау, процессордың уақытын пайдалану);
- Бағдарламаны тестілеу құралдарының болуы**.**

2.2.1 IntelliJ IDEA

IntelliJ IDEA-JetBrains жасаған Java, JavaScript, Python сияқты көптеген бағдарламалау тілдеріне арналған бағдарламалық жасақтаманы әзірлеудің интеграцияланған ортасы. IntelliJ IDEA-бұл платформалық өнім. Ол Java-да жұмыс істейді, яғни оны Windows, OSX және Linux-та іске қосуға болады. Сіз Oracle JVM дистрибутивін немесе OpenJDK (OpenJDK 6-дан басқа) қолдана аласыз.

IntelliJ IDEA – бұл Java тіліне арналған IDE ғана емес, басқа тілдерге арналған ондаған плагиндер бар, олардың ішінде Clojure, Scala, Kotlin, Go, HTML, JavaScript және басқалары бар. Бұл плагиндердің кейбірін JetBrains, ал басқаларын қауымдастық жасаған.

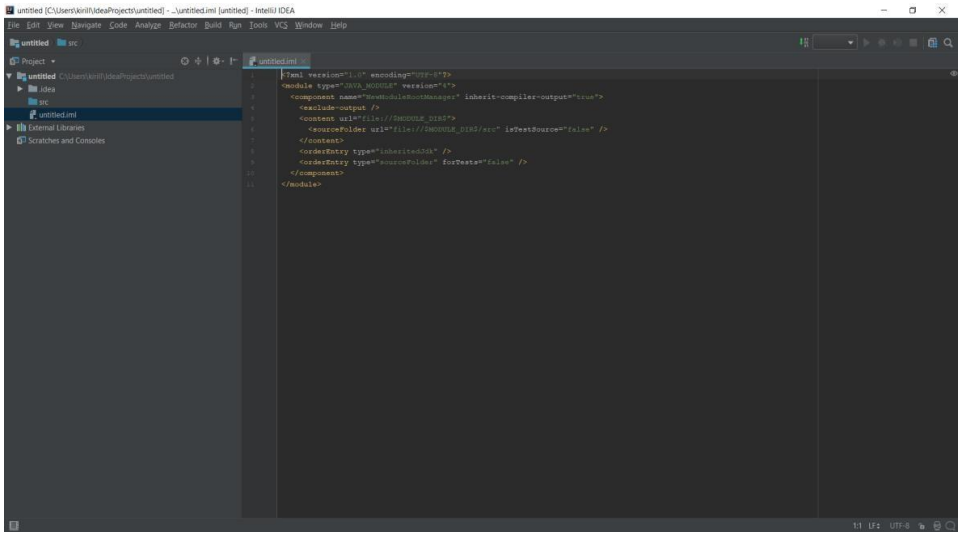
Сіз HTML және JavaScript көмегімен жасалған және Java-мен ешқандай байланысы жоқ веб-қосымшаларды жасау үшін IntelliJ IDEA қолдана аласыз.

Мұндай жағдайларда WebStorm пайдалану ұсынылады (егер сізге JVM технологиясы қажет болмаса).

Тілдерден басқа, Ол көктем, Android, Java EE және басқалары сияқты көптеген технологиялар мен шеңберлерді қолдайды.

IDE кеңейтімділігі-бұл плагин репозиторийінен жүктеуге немесе өзіңіз жасай алатын көптеген плагиндер.

Бұл дипломдық жоба OpenCV әзірлеу тобының IDE таңдау ұсыныстарына байланысты осы IDE-ді қолданбаған.



2.1-сурет IntelliJ IDEA негізгі терезесі

2.2.2 NetBeans IDE

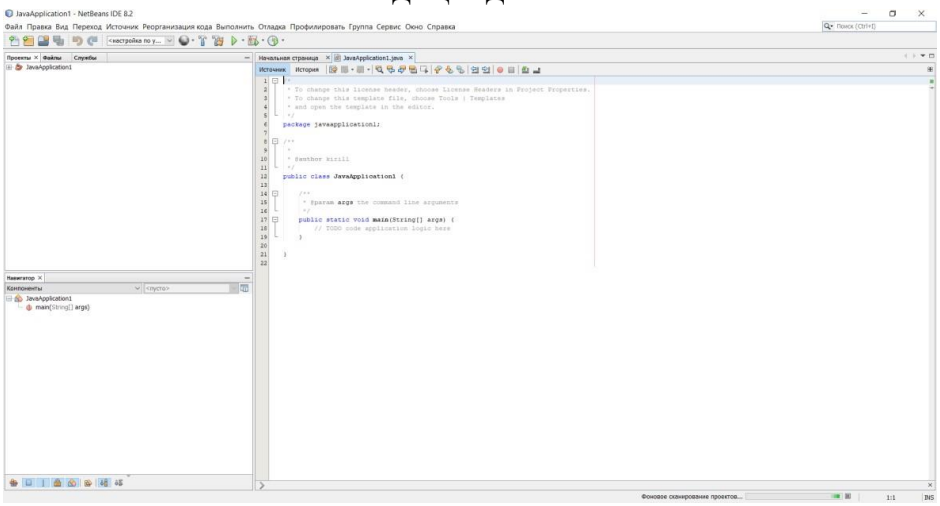
NetBeans-Java үшін интеграцияланған даму ортасы (IDE). NetBeans модуль деп аталатын модульдік бағдарламалық компоненттер жиынтығынан қосымшаларды жасауға мүмкіндік береді. NetBeans Microsoft Windows, macOS, Linux және Solaris жүйелерінде жұмыс істейді. Java-да дамудан басқа, оның PHP, C, C ++, HTML5, Javadoc және Javascript сияқты басқа тілдерге арналған кеңейтімдері бар.

Ide NetBeans – бұл ашық бастапқы коды бар интеграцияланған даму ортасы. Ide NetBeans Java Қосымшаларының барлық түрлерін (Java SE (JavaFX қоса), Java ME, web, EJB және Мобильді қосымшаларды) қораптан тыс әзірлеуді қолдайды. Басқа функцияларға құмырсқаларға негізделген дизайн жүйесі, Maven қолдауы, рефакторинг, нұсқаны басқару (CVS, Subversion, Git, Mercurial және Clearcase қолдауы) кіреді.

Барлық IDE функциялары модульдермен қамтамасыз етіледі. Әр модуль Java тілін қолдау, CVS және SVN нұсқаларын өңдеу немесе басқаруды қолдау сияқты нақты анықталған функцияларды ұсынады. NetBeans-та Java-ны қораптан шығаруға қажетті барлық модульдер бар, сондықтан сіз бірден дамуды бастай аласыз. Модульдер сонымен қатар NetBeans-ты кеңейтуге мүмкіндік береді. Басқа бағдарламалау тілдерін қолдау сияқты жаңа мүмкіндіктерді қосымша модульдерді орнату арқылы

қосуға болады. Мысалы, Sun Studio, Sun Java Studio Enterprise және Sun Microsystems-тің Sun Java Studio Creator Ide NetBeans-қа негізделген.

Бұл дипломдық жоба OpenCV әзірлеу тобының IDE таңдау ұсыныстарына байланысты осы IDE-ді қолданбаған.



2.2- сурет NetBeans IDE негізгі терезесі

2.2.3 Eclipse IDE

Eclipse ең алдымен кеңейтімдерді дамыту платформасы ретінде қызмет етеді, сондықтан ол танымал болды: кез-келген әзірлеуші Eclipse-ді өз модульдерімен кеңейте алады. Қазірдің өзінде Java әзірлеу құралдары (JDT), QNX инженерлері IBM-мен бірлесіп әзірлеген C / C ++ (CDT) әзірлеу құралдары, сондай-ақ Ada (GNATbench, Hibachi), COBOL, FORTRAN, PHP, X10 (X10DT) тілдеріне арналған құралдар бар . әр түрлі әзірлеушілерден. Көптеген кеңейтімдер Eclipse ортасын мәліметтер базасымен, қолданба серверлерімен және басқалармен жұмыс істеуге арналған диспетчерлермен толықтырады.

Eclipse JDT (Java Development Tools) – топтық дамуға бағытталған ең танымал модуль: орта нұсқаларды басқару жүйелерімен біріктірілген-CVS, GIT негізінен тарату, басқа жүйелер үшін (мысалы, Subversion, MS SourceSafe) плагиндер бар. Сондай-ақ, IDE мен тапсырмаларды басқару жүйесі (қателер) арасындағы байланысты қолдайды. Негізгі пакетке Bugzilla қате трекерін қолдау кіреді, сонымен қатар басқа трекерлерді (Trac, Jira және т.б.) қолдау үшін көптеген кеңейтімдер бар. Бұл ақысыз және сапалы болғандықтан, Eclipse көптеген ұйымдарда қосымшаларды әзірлеудің корпоративті стандарты болып табылады.

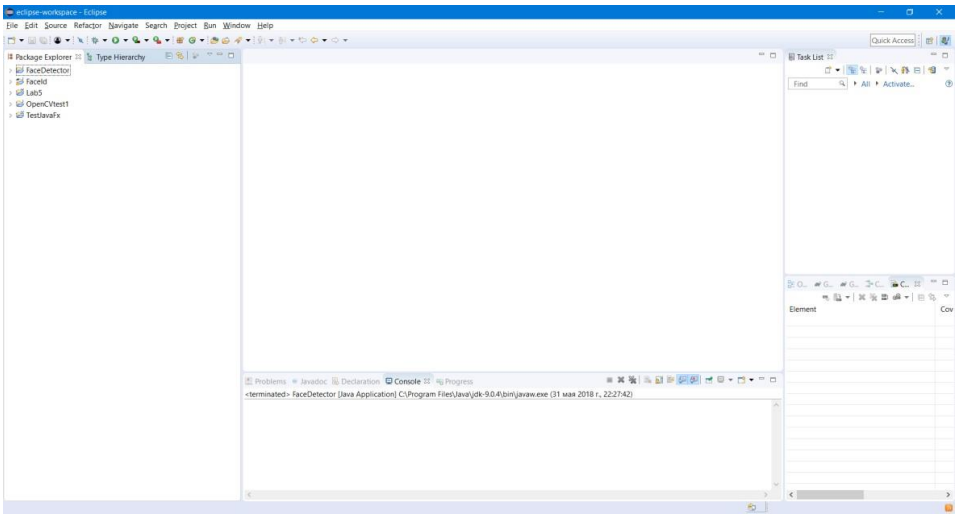
Eclipse Java-да жазылған, сондықтан ол барлық жалпы платформаларға арналған SWT кітапханасынан басқа платформаға тәуелді емес (төменде қараңыз). SWT кітапханасы Java Swing стандартты кітапханасының орнына қолданылады. Ол жылдам және табиғи пайдаланушы тәжірибесін қамтамасыз ету үшін негізгі платформаға (амалдық жүйеге) толығымен сүйенеді, бірақ

кейде әртүрлі платформаларда қосымшалардың үйлесімділігі мен тұрақтылығына қиындық тудырады.

Eclipse IDE Java Ide, git клиенті, XML редакторы, Mylyn, Maven және Gradle интеграциясы сияқты негізгі құралдарды қамтиды, Eclipse платформасы еркін лицензиялардың біріне, бастапқыда жалпы мемлекеттік лицензияға және қазір Eclipse қоғамдық лицензиясына сәйкес пайдалануға лицензияланған. Бұл IBM лицензиялары FSF-пен тегін және мақұлданған, бірақ олар қоғамдық игілікті GPL сияқты ұрлықтан қорғамайды. Бұл православиелік бизнеске Eclipse жабық плагиндерінде білімдерін жасыруға, ноу-хауды (ноу-хау, өндіріс құпиясы) қорғауға және басқа адамдардан бәсекелестік артықшылық алуға мүмкіндік беретін ымыралы лицензиялар.

Eclipse ортасы үшін бірқатар ақысыз және коммерциялық модульдер бар. Бастапқыда Орта Java тілі үшін жасалған, бірақ қазір басқа тілдерді қолдау үшін көптеген кеңейтімдер бар: C ++, Fortran, Perl, PHP, JavaScript, Python, Ruby, 1C V8.

Бұл дипломдық жоба үшін Java үшін Eclipse IDE әзірлеу ортасы таңдалды.



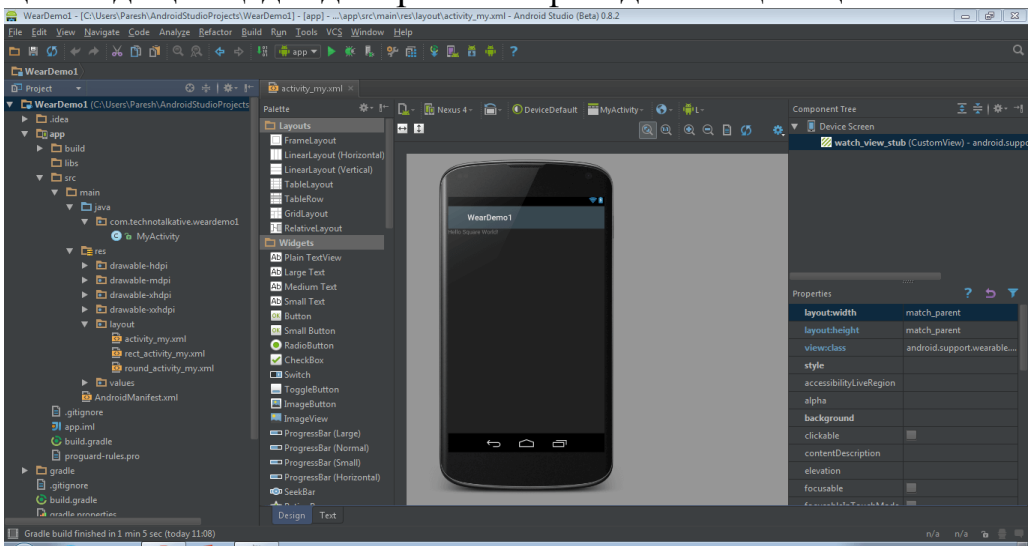
2.3-сурет Eclipse IDE негізгі терезесі

2.2.4 Android Studio бағдарламалау ортасы

Android Studio – бұл Android үшін ресми бағдарлама жасау ортасы. Негізінен, Android Studio-плагиндері бар әйгілі Java IDE IntelliJ идеясы. Ең алдымен, сізде JDK (Java Development Kit) орнатылғанына көз жеткізу керек. Бұл Java-ны дамытудың міндетті шарты және Android-ті дамыту Java-да орындалатындықтан, Android-ті де дамыту керек. Сіз JDK-ны осы сілтемеден жүктей аласыз (Ресми сайт, әрине, JDK тегін, ең жаңа JDK таңдаңыз, және бұл JRE емес, JDK!

Енді Android Studio-ны орнатуға көшейік. 1) JDK жүктеңіз, Oracle веб-сайтына өтіңіз, JDK төте жолының астындағы Жүктеу түймесін басыңыз, келісімді қабылдаңыз және Windows x64 (64 биттік сәулет жағдайында)

немесе Windows x86 үшін Java SE Development Kit 7 жүктеңіз; 2) Java орнатыңыз алдыңғы қадамда жүктелген файлды іске қосыңыз.



2.4-сурет Android studio негізгі терезесі

2.3 Қолданылған технологиялардың сипаттамасы

2.3.1 Android SDK мобильді қосымшаларды әзірлеудің әмбебап құралы

Android SDK – бұл Android операциялық жүйесіне арналған мобильді қосымшаларды әзірлеуге арналған әмбебап құрал. Басқа код редакторларынан ерекшелігі-бастапқы кодтарды тексеруге және күйін келтіруге, қосымшаның жұмысын Android ОЖ-нің әртүрлі нұсқаларымен үйлесімділік режимінде бағалауға және нәтижені нақты уақыт режимінде бақылауға мүмкіндік беретін кең функционалды қосымша болып табылыады. Ол көптеген мобильді құрылғыларды қолдайды, олардың ішінде: ұялы телефондар, планшеттік компьютерлер, смарт-көзілдіріктер (соның ішінде Google Glass), Android ОЖ-де борттық компьютерлері бар заманауи автомобильдер, кеңейтілген функционалдығы бар теледидарлар, қол сағаттарының арнайы түрлері және басқа да көптеген мобильді гаджеттер.

2014 жылдың соңына дейін Eclipse IDE Android Development Tools (ADT) плагинімен толықтырылған код редакторы ретінде қолданылды. 2015 жылы Android Studio әзірлеу жинағы шығарылды (Google IntelliJ ide технологиясын қолдана отырып жасаған), ол негізгі болды. 2015 жылдың соңында ADT ескі деп саналып, ал Android Studio Android қосымшаларын әзірлеудің негізгі жүйесіне айналды. IDE интеграциясынан басқа, біз Java және XML файлдарын жасау үшін үшінші тараптың мәтіндік редакторларын қолдана аламыз және жобаларды құру, құрастыру және жөндеу үшін пәрмен жолының утилиталарын (Java Development Kit және Apache Ant қажет) қолдана аламыз. Қосылған Android құрылғыларын басқару утилиталары

қосымшаларды қайта жүктеу және орнату үшін де қол жетімді: fastboot және adb (Android Debug көпірі).

SDK-де Android платформасының ескірген нұсқаларының бөліктері болуы мүмкін, әзірлеушілер ескірген телефондар мен планшеттерге арналған қосымшаларды әзірлеуді жалғастыруға дайын. Кейбір Даму құралдары бөлек жүктелетін DLC түрінде келеді.

Android қосымшалары толығымен пакеттер болып табылады .apk және орнатқаннан кейін / data / app каталогында сақталады. Ішінде APK код файлдарын қамтиды .dex(Dalvik-те орындалатын байт коды), ресурстық файлдар және т. б.

2.3.2 OpenCV және тұлғаны тану әдістері

Қосымшаны жазу барысында ақысыз және ашық бастапқы коды бар нарық зерттелді. Ең дұрыс шешім-әзірлеушілердің белсенді қолдауы мен кескіндерді өңдеудің кең мүмкіндіктері арқасында OpenCV кітапханасын пайдалану болды. Бағдарламаның соңғы нұсқасы статистика жинау модулінен және тану модулінен тұрады.

Тану модулі Java-да OpenCV кітапханасын қолдана отырып жазылған. Оны жүзеге асыру ұшін қолданылды :

- Бет пен көзді іздеуге арналған Хаар каскадтары (мұрын);
- OpenCV кітапханасынан суреттерді өңдеу әдістері;
- Нысандарды есептеу және салыстыру үшін жергілікті екілік үлгілер.

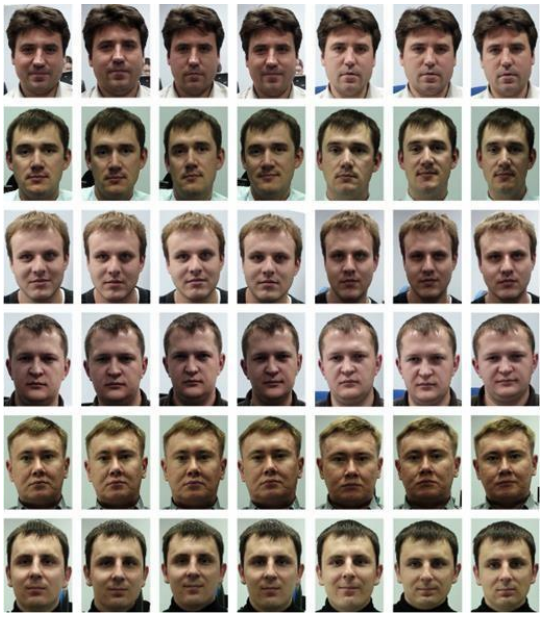
Кіріс ретінде командалық жолдың дәлелдері қолданылды (бірінші кескінге жол, екінші кескінге жол). Шығу кезінде біз белгілі бір қашықтықты жүріп өттік. Бұл қашықтық неғұрлым аз болса, суреттердің "ұқсастығы" соғұрлым үлкен болады. Жобаны жинау үшін maven қолданылды.

Статистика модулі Java-да жазылған. Ол кіріс ретінде 3 аргументті қабылдайды: тану модуліне жол, фотосуреттермен үлгі жолы, егжей-тегжейлі статистиканы жазу үшін файл жолы. Бұл бағдарлама әр фотосуретті әр сынақ үлгісімен салыстыру арқылы тану модулін іске қосады. Нәтижесінде біз егжей-тегжейлі статистика мен жалпы статистика бар csv файлын аламыз.

Кітапханада 2500-ден астам оңтайландырылған алгоритмдер бар, олар классикалық және заманауи компьютерлік көру және машиналық оқыту алгоритмдерінің толық жиынтығын қамтиды. Бұл алгоритмдерді бетті анықтау және тану, нысандарды анықтау, бейнелердегі адам әрекеттерін жіктеу, камера қозғалысын бақылау, қозғалатын заттарды бақылау, 3D объект модельдерін алу, стерео камералардан 3D нүкте бұлттарын жасау, барлық көріністердің жоғары ажыратымдылықтағы кескінін алу үшін суреттерді тігу, кескін дерекқорынан ұқсас суреттерді іздеу, стерео камералармен жасалған суреттерден қызыл көзді алып тастау үшін пайдалануға болады.

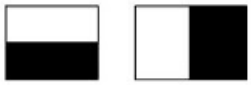
Жыпылықтаңыз, көздің қимылын бақылаңыз, ландшафттарды таныңыз және кеңейтілген шындыққа қабаттасу үшін маркерлер орнатыңыз және тағы басқалар. OpenCV-де 47000-нан астам қауымдастық қолданушылары бар және шамамен 14 миллионнан астам Жүктеу бар. Кітапхананы компаниялар, ғылыми-зерттеу топтары және мемлекеттік мекемелер кеңінен қолданады. Хаар каскадты классификаторларын қолдана отырып, нысандарды анықтау-бұл Пол Виоле мен Майкл Джонс 2001 жылы "қарапайым функциялардың күшейтілген каскадын қолдана отырып, нысандарды жылдам анықтау"мақаласында ұсынған нысандарды анықтаудың тиімді әдісі. Бұл оқыту механизміне негізделген тәсіл, каскадты функция көптеген оң және теріс бейнелерден үйренеді. Содан кейін ол басқа суреттердегі нысандарды анықтау үшін қолданылады.

Бастапқыда алгоритмді үйрену үшін көптеген оң кескіндер қажет (бет суреттері) (суретті қараңыз. 1.5) және теріс кескіндер (бет-әлпетсіз суреттер) классификаторды оқыту үшін.

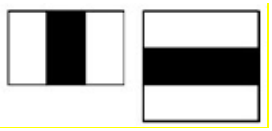


2.5-сурет суреттердің оң мысалдары

Содан кейін олардан функцияларды бөліп алу керек. Ол үшін Haar функциялары қолданылады. Әр функция - бұл ақ тіктөртбұрыш астындағы пиксельдер қосындысын қара тіктөртбұрыш астындағы пиксельдер қосындысынан шығару арқылы алынған жалғыз мән болып табылады. (2.6-2.8 - суретті қараңыз)



2.6-сурет шекаралық функциялар



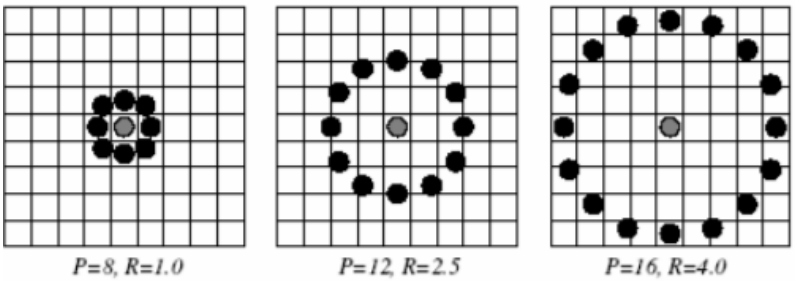
2.7-сурет сызықтық функциялар

2.8-сурет төрт төртбұрыштың функциялары

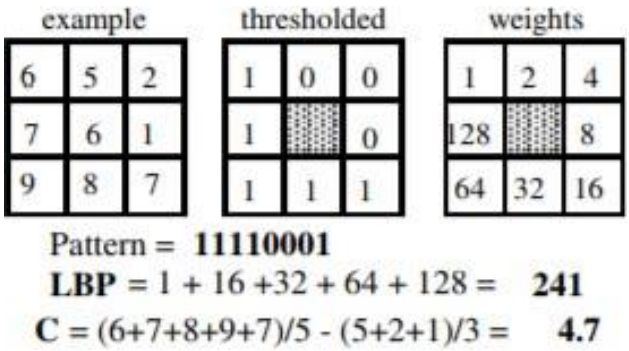
Адамның толық келбетін анықтайтын сонымен қатар бақылайтын келесі әдіс – бұл Жергілікті екілік шаблондарды қолдану болып табылады. Жергілікті екілік өрнектер (LBP) - бұл компьютердің көруіндегі текстураны жіктеу үшін қолданылатын қарапайым оператор. LBP операторын алғаш рет Т.Ожала 1996 жылы ұсынды. LBS - екілік түрдегі кескін пикселінің көршілігін сипаттау арқылы жұмыс істейді. Сурет пикселіне қолданылатын LBS операторы сегіз көршілестік пикселді пайдаланады, орталық пикселді шекті мән ретінде қабылдайды. Бұл кескіннің әр пикселін суреттегі көрші пикселдердің қарқындылығына байланысты екілік сан ретінде көрсететін тиімді оператор. Осылайша, пиксельдің айналасын сипаттайтын сегіз биттік екілік код алынады.

LBP операторы суреттегі объектіні (мысалы, тұлға) іздеу үшін, сондай-ақ осы объектіні белгілі бір сыныпқа (тексеру, эмоцияны тану, бет жынысы) жататындығын тексеру үшін пайдаланылуы мүмкін.

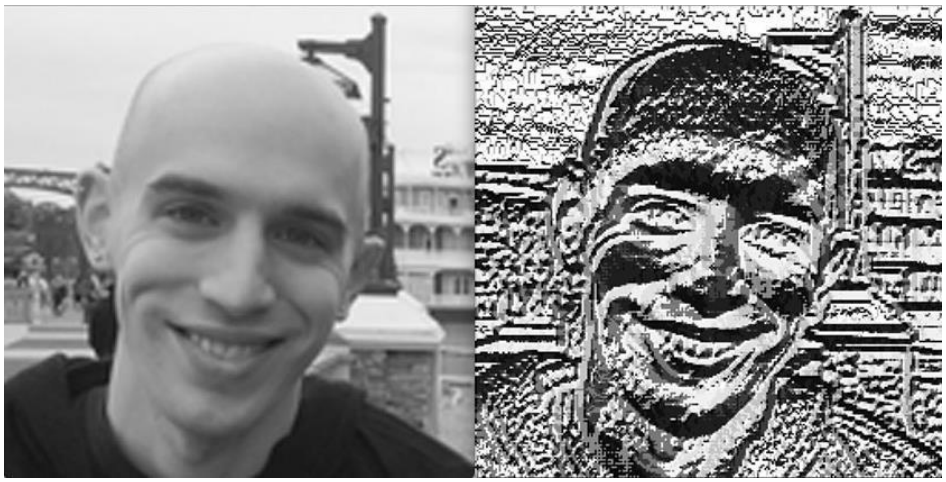
Бұл оператор есептеуіш тиімді, өйткені ол тек бүтін арифметикамен жұмыс істейді (бұл кейбір тапсырмаларда нақты уақыттағы өнімділікке қол жеткізуге мүмкіндік береді), сонымен қатар әртүрлі жарық жағдайларында түсіруден туындаған кескін жарықтығының өзгеруіне инвариантты.



2.9-сурет құрылымды анықтау және жергілікті екілік үлгіні (LBP) есептеу үшін қолданылатын ортаның үш мысалы)



2.10-сурет LBP бастапқы кодын есептеу және контраст өлшемдері

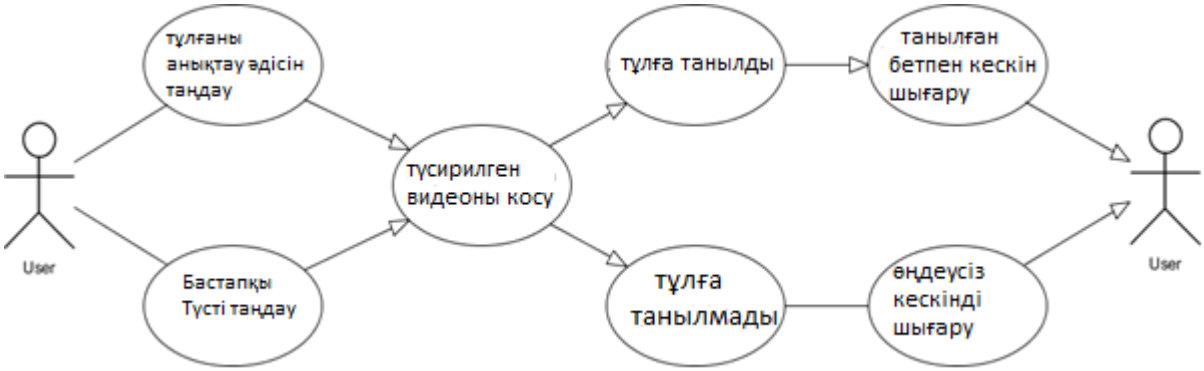


2 .11-сурет LBP көмегімен бетті өңдеудің мысалы

LBP-дің осы түпнұсқалық іске асырылуының басты артықшылығы-біз кескіннің өте ұсақ бөлшектерін түсіре аламыз. Алайда, бөлшектерді осындай кішкентай масштабта түсіру мүмкіндігі алгоритмнің ең үлкен кемшілігі болып табылады-біз бөлшектерді әр түрлі масштабта түсіре алмаймыз, тек 3x3 пиксель бекітілген масштаб.

3 Мобильді қосымша құруды тәжірибе жүзінде жүзеге асыру

3.1 UML диаграммасы

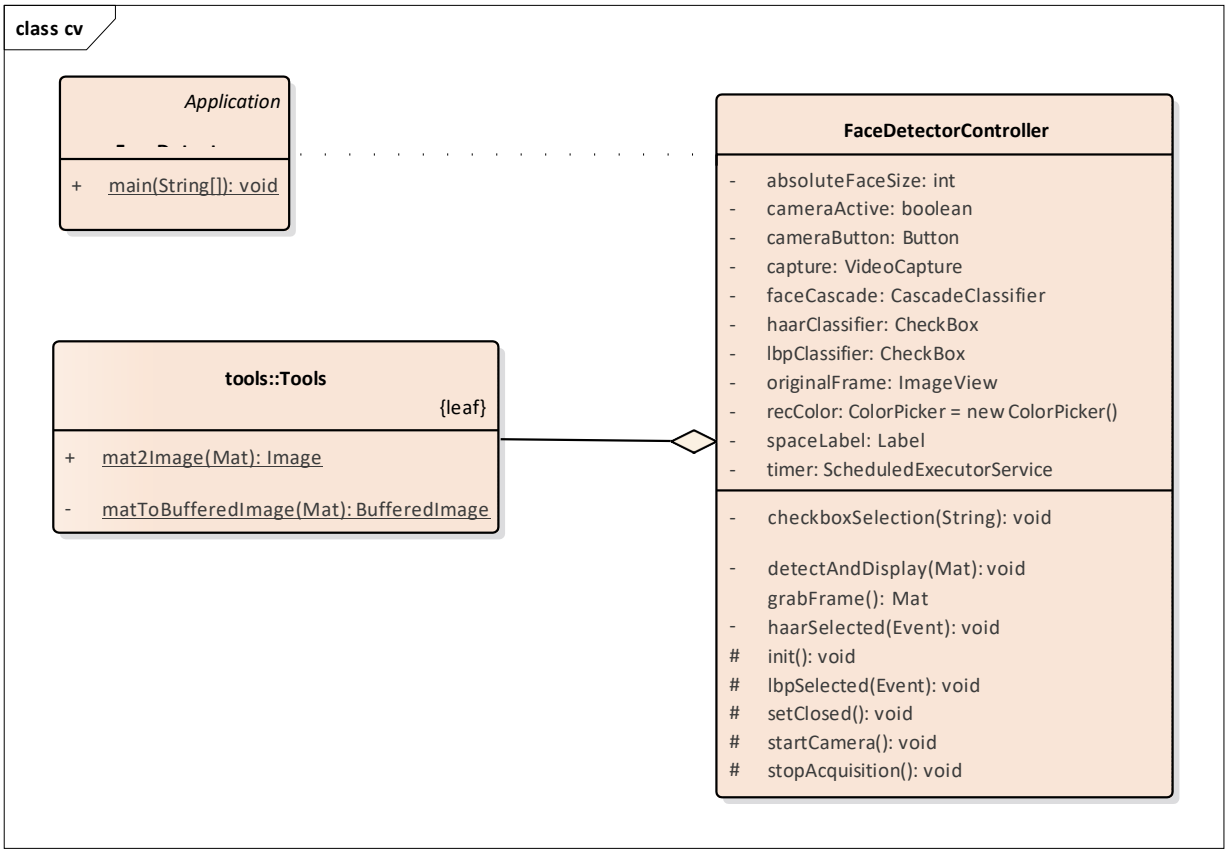


3.1-сурет пайдалану диаграммасы

Сынып диаграммасы-объектіге бағытталған модельдеудің негізгі құрылыс блогы. Ол жүйелік қосымшаны жалпы тұжырымдамалық модельдеу үшін және модельдерді бағдарламалық кодқа аударуды егжей-тегжейлі модельдеу үшін қолданылады. Сынып диаграммаларын деректерді модельдеу үшін де қолдануға болады. Сынып диаграммасындағы сыныптар негізгі элементтерді де, қосымшадағы өзара әрекеттесуді де, бағдарламалануы керек сыныптарды да білдіреді.

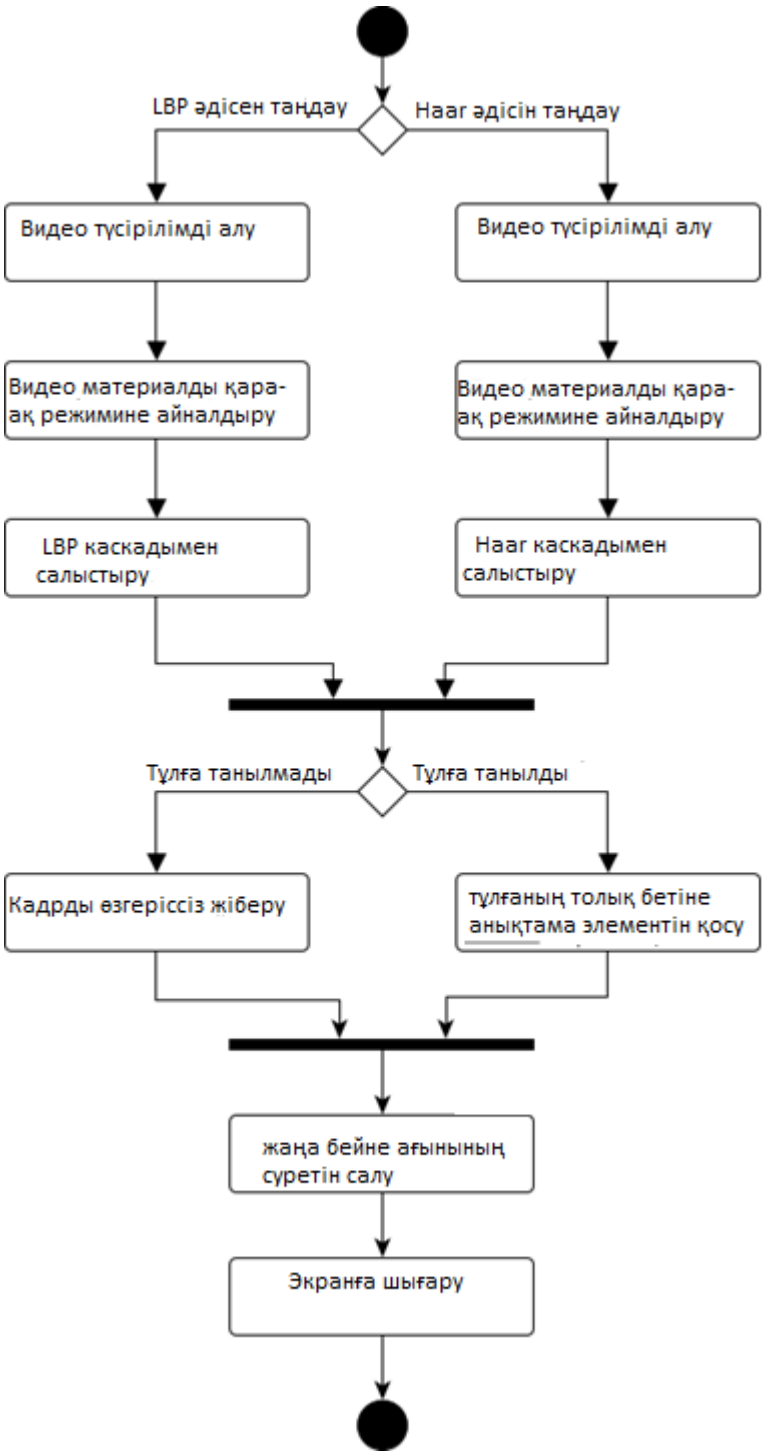
Диаграммада сыныптар үш блоктан тұратын квадрат фигуралармен ұсынылған:

- Жоғарғы блокта сынып атауы бар. Ол қалың және ортасында, алдымен бас әріптермен басылады;
- Орта блокта сынып атрибуттары бар. Олар сол жақта туралanған және бірінші кіші әріптен тұрады;
- Төменгі блокта сынып орындай алатын операциялар бар. Олар сондай-ақ сол жақта туралanған және бірінші әріп кіші болады.



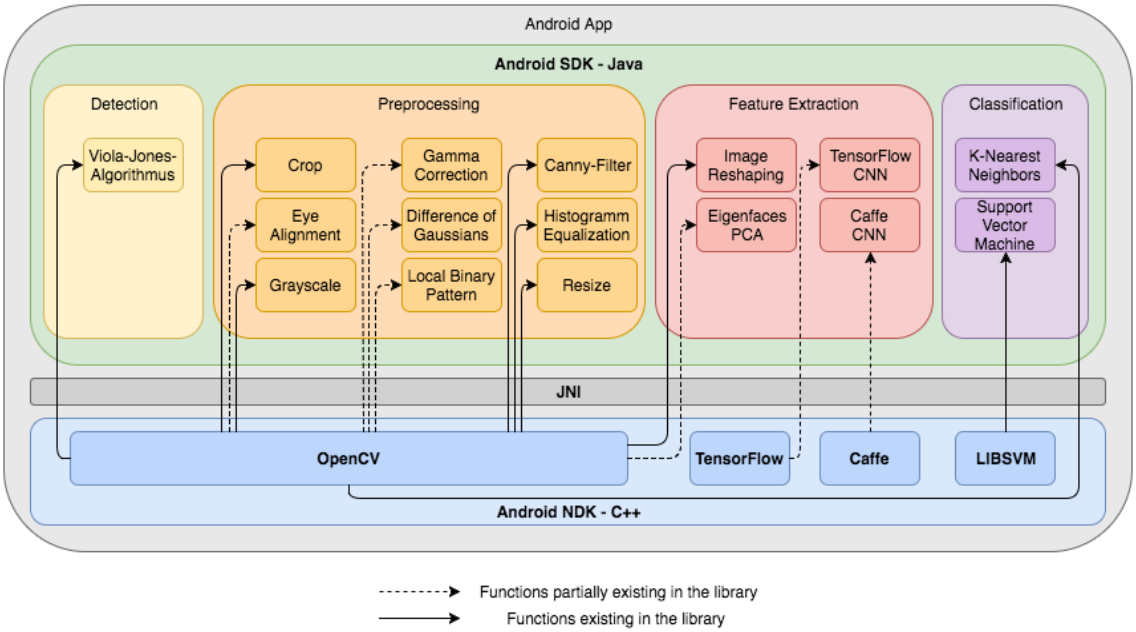
3.2 -сурет Сыныптар диаграммасы

Әрекет диаграммалары-таңдау, Итерация және параллелизмді қолдайтын қадамдық және жұмыс процестерінің графикалық көріністері. Бірыңғай модельдеу тілінде іс-қимыл диаграммалары есептеу және ұйымдастырушылық процестерді (мысалы, жұмыс процестері), сондай-ақ тиісті әрекеттермен қиылысатын мәліметтер ағынын модельдеуге арналған. Әрекет диаграммалары негізінен басқарудың жалпы ағынын көрсетсе де, олар бір немесе бірнеше деректер қоймасындағы әрекеттер арасындағы деректер ағынын көрсететін элементтерді қамтуы мүмкін.



3.3-сурет белсенділік диаграммасы

Кез келген жобаның архитектурасы болады. Android қосымшаларда соның қатарына кіреді. 3.4 – суретте көріп отырғандай, жобаның архитектурасында жүзеге асырылатын барлық қадам көрсетіледі.



3.4-сурет Жобаның архитектуралық құрылымы

3.2 Мобильді қосымша құруға дайындық шаралары

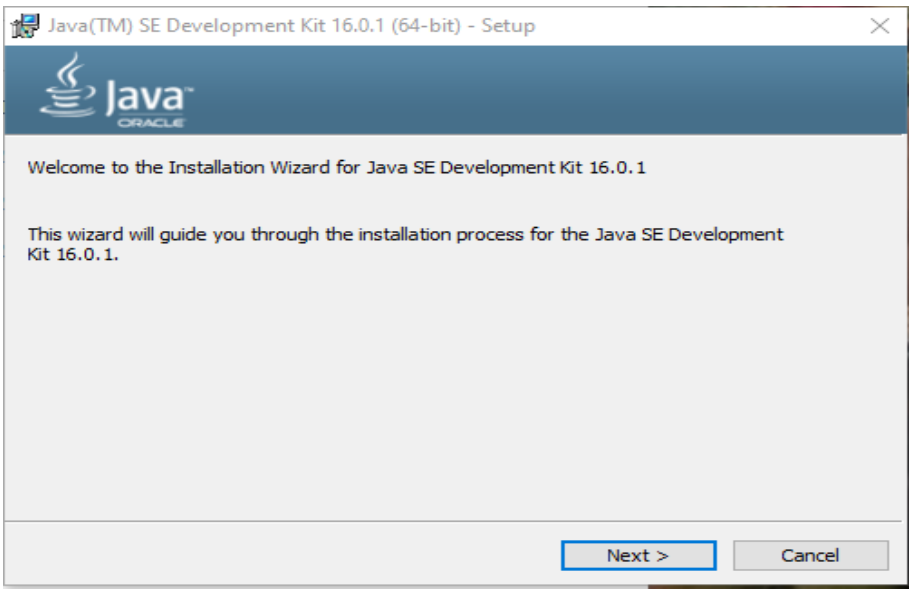
Бағдарламаны құруға дайындық шаралары-қажетті бағдарламалық жасақтаманы, плагиндерді, кеңейтімдерді, кітапханаларды орнату және оларды дұрыс орнату.

Қосымшаны құру үшін ыңғайлы жұмыс Java әзірлеушілеріне арналған Eclipse IDE сияқты бағдарламалардың арқасында жүзеге асырылады (нұсқа: Oxygen 4.7.2 x64), Android SDK плагины, Java 16.

3.2.1 Java программалау тілін орнату

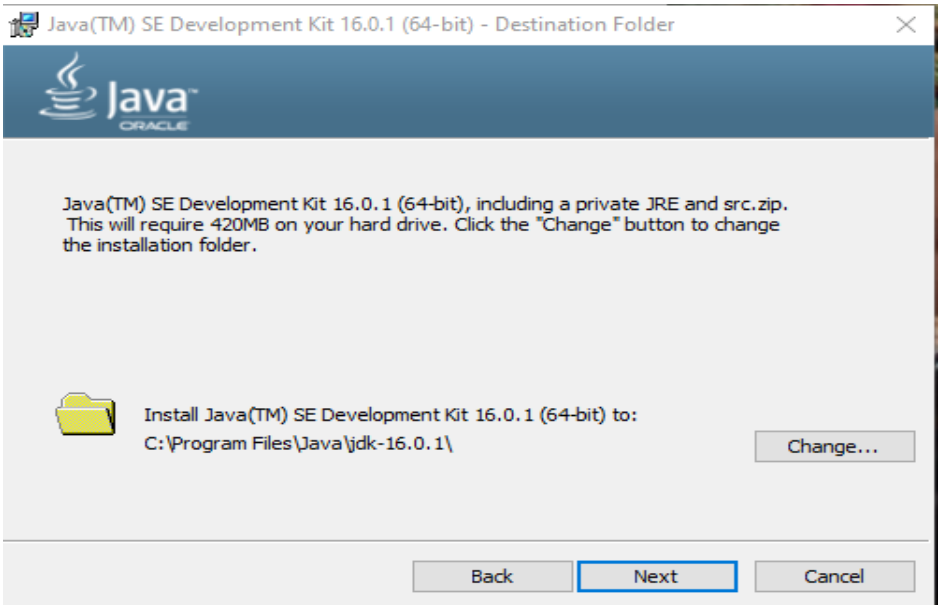
Java SE Development Kit 16 орнату Oracle ресми веб-сайтынан осы орнату бумасын жүктеуден басталады.

Бірінші орнату терезесінде пакеттің осы нұсқасы туралы жалпы ақпарат бар (суретті қараңыз. 2.4).



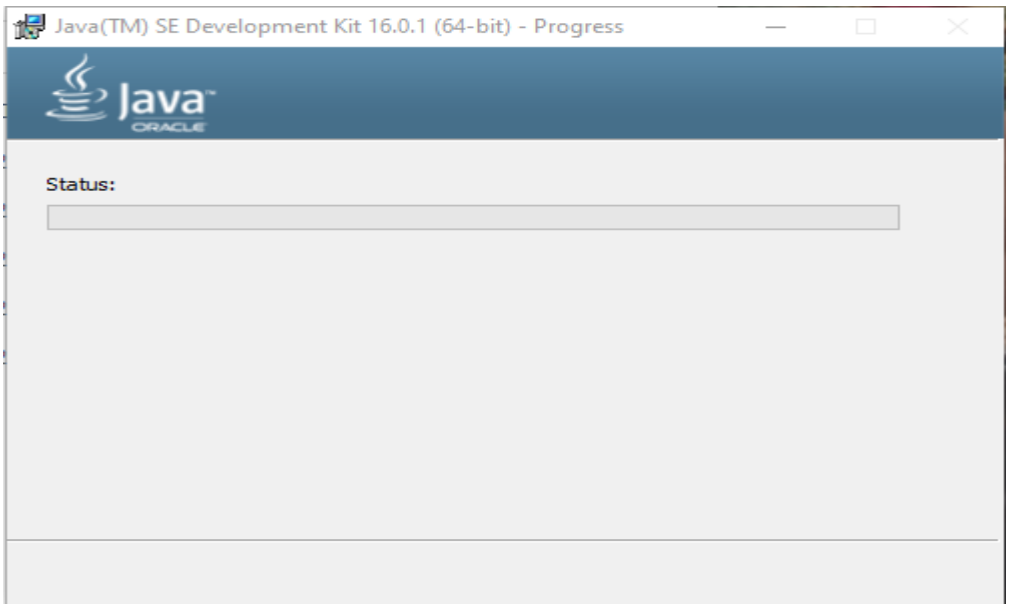
3.5-сурет Java бастапқы орнату терезесі

Осыдан кейін сіз осы жинаққа кіретін жұмыс үшін қажетті құралдарды таңдауыңыз керек.



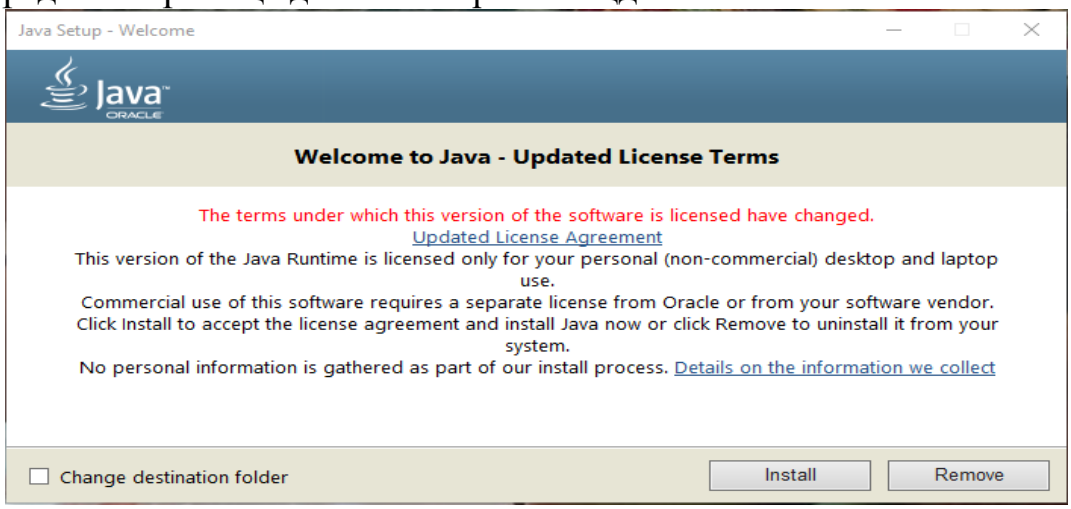
3.6-сурет Тиісті компоненттерін таңдау

Содан кейін шебер таңдалған компоненттерді орнатқанша күтуіңіз керек, сонымен қатар жүйеде осы компоненттерді тіркеңіз.

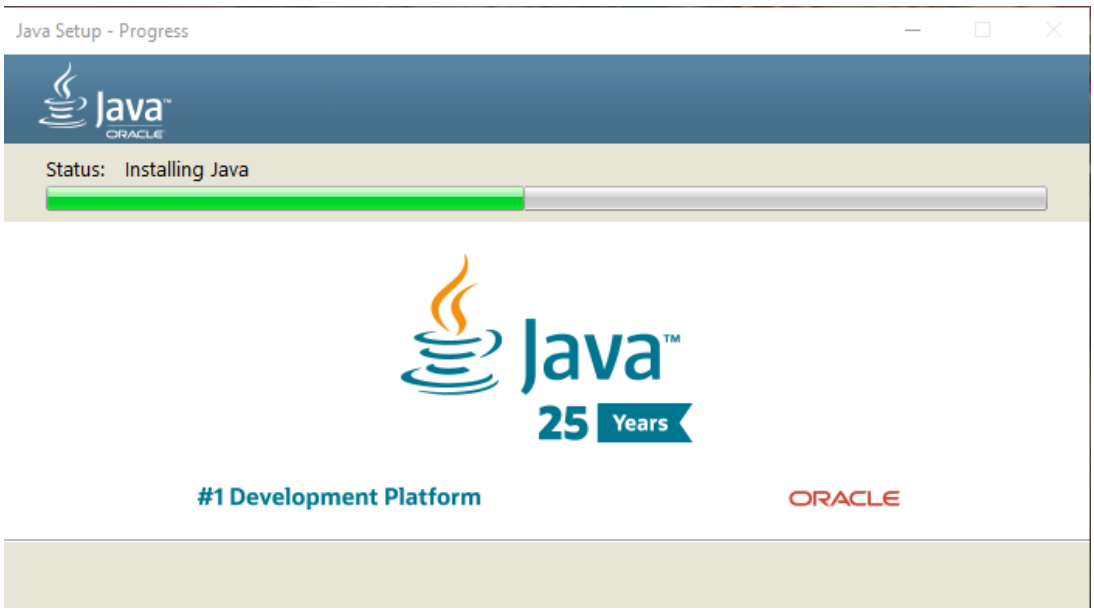


3.7-сурет Орнату процесі

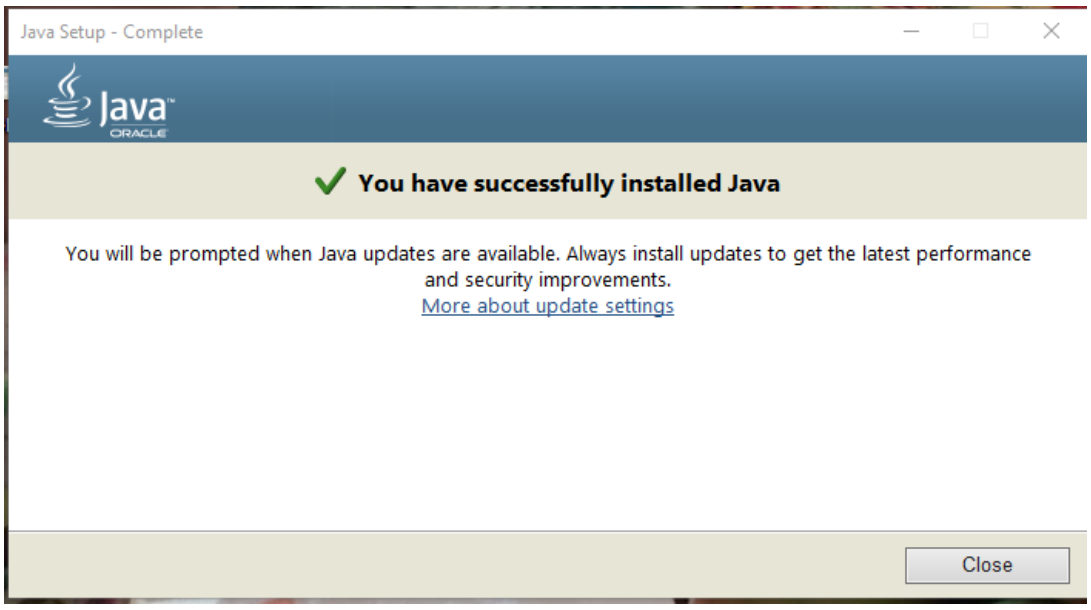
Осыдан кейін сіз осы пакетке кіретін JRE жиынтығын орнатуыңыз керек. Мұнда сіз жүйенің әдепкі JRE орнын таңдай аласыз.



3.8-сурет JRE орнатудың бастапқы терезесі



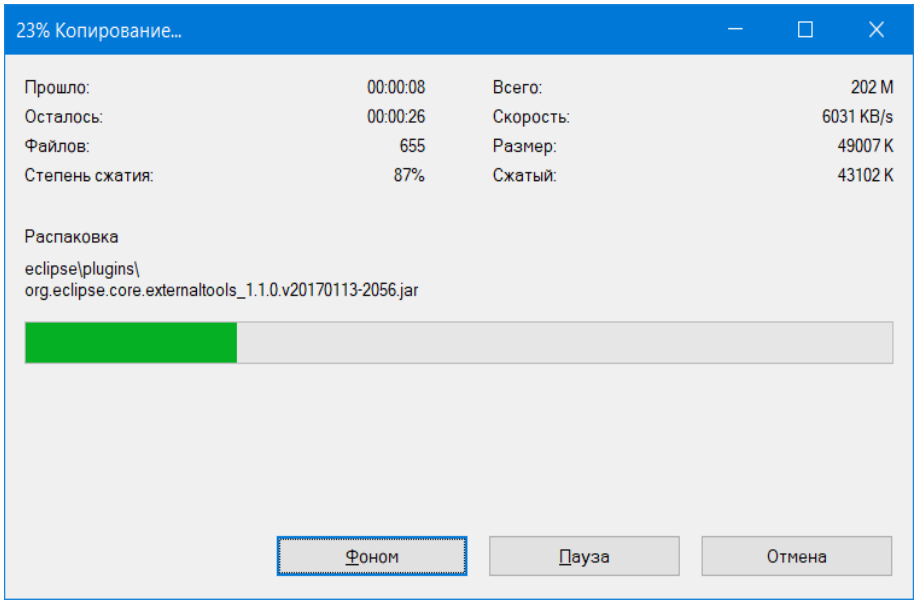
3.9-сурет JRE орнату процесі



3.10-сурет Орнатуды аяқтау терезесі

2.2.2 Eclipse әзірлеу ортасын орнату және баптау

Eclipse IDE - ді әзірлеуші іске қосу үшін барлық қажетті файлдарды қамтитын zip кескіні ретінде ұсынады. Сізге барлық IDE-ді ыңғайлы қалтаға шығару керек.



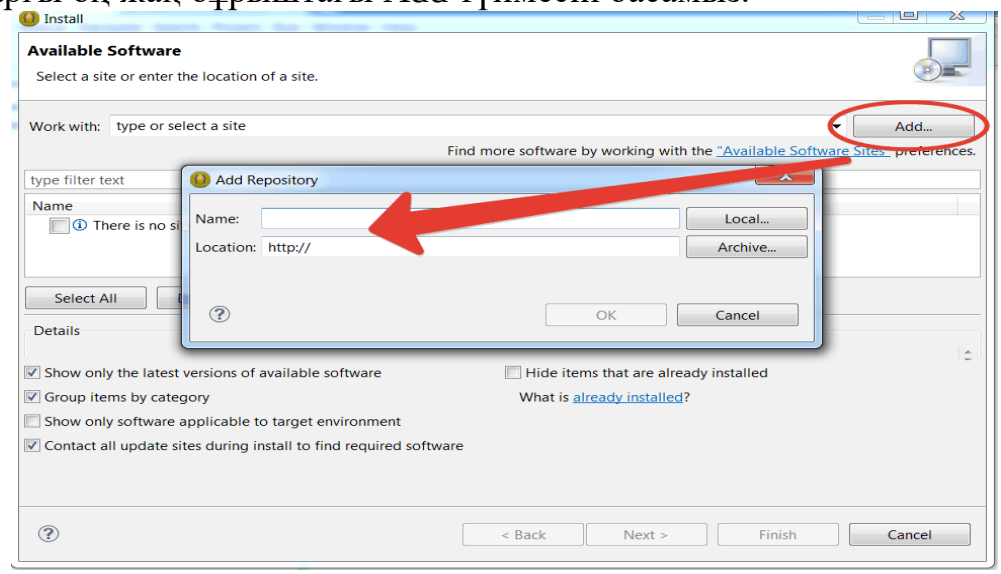
3.11 - сурет-Eclipse IDE орауыштан шығару процесі

Android қосымшаларын әзірлеу үшін Eclipse Android Development Tools (ADT) деп аталатын Eclipse IDE үшін арнайы плагин ұсынады. Бұл плагин Android қосымшалары үшін қуатты интеграцияланған даму ортасы болып табылады. Бұл Eclipse-ге жаңа Android жобаларын тез құруға, қолданба интерфейсін құруға, бағдарламаны күйге келтіруге және тарату үшін қосымшалар пакеттеріне (APK) қол қоюға және экспорттауға мүмкіндік береді.

Eclipse үшін ADT плагинін жүктеп аламыз:

Eclipse іске қосамыз, содан кейін Негізгі мәзірден таңдаймыз Help > Install New Software..

Жоғарғы оң жақ бұрыштағы Add түймесін басамыз.



3.12 -сурет ADT плагинын орнату

Репозиторий диалог терезесінде Add өрісіне «ADT Plugin» және Location өрісіне келесі адресті енгізіңіз:

https://dl-ssl.google.com/android/eclipse/

Android SDK жүйесін баптау

Eclipse қайта іске қосылғаннан кейін сіз Android SDK каталогының орнын көрсетуіңіз керек:

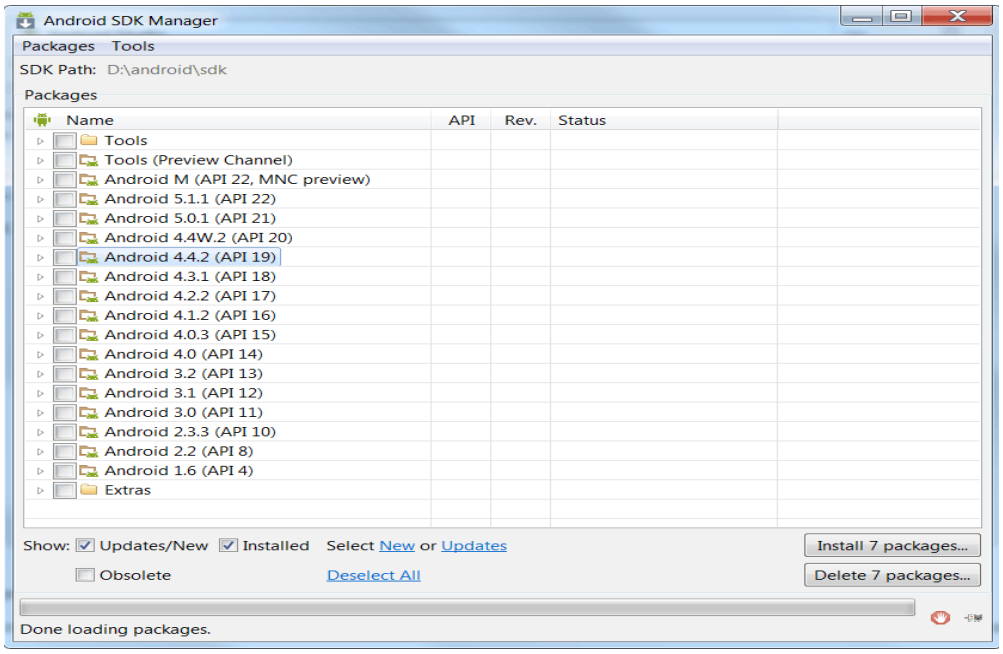
Сәлемдесу терезесінде Бар SDK-ді пайдалану таңдаңыз.

Browse (Шолу) батырмасын шертіп, SDK каталогының орнын таңдаңыз (SDK-ны жүктегенде жолды көрсеткен жерде).

Келесі түймешігін басыңыз.

Eclipse IDE Android қосымшаларын әзірлеу үшін орнатылған, бірақ сіз өзіңіздің ортаңызға жаңа SDK платформаларын және Android платформасын қосуымыз керек. Осы пакеттерді алу үшін SDK менеджерін іске қосып, жүктелетін қажетті платформалар мен бумаларды таңдаймыз.

Eclipse ішінде мәзірден Window> Android SDK Manager ашамыз



3.13-сурет Android SDK Manager терезесі

Мұнда біз жүктеуге, жаңартуға немесе жоюға болатын SDK компоненттері көрсетілген.

Алдымен Tools қалтасы келеді-онда Android-ті дамыту үшін қажетті утилиталар бар. Төменде Android нұсқаларының тізімі берілген. Төменгі жағында әлі де қосымша кітапханалар бар Extras қалтасы бар.

Тізімдегі әр компоненттің оң жағында оның күйін көруге болады: Installed - орнатылған, Notinstalled - орнатылмаған, Update қол жетімді - жаңарту қол жетімді. Маған автоматты түрде кейбір компоненттерді жаңарту және орнату ұсынылады, оларда құсбелгілер бар. Барлық құсбелгілерді, содан кейін жаңартуларды жою үшін тек жаңартуды қажет ететін элементтерді

таңдау үшін төмендегі таңдаудан бас тарту сілтемесін нұқыңыз. Біз бәрін сол күйінде қалдырсақ, біз ештеңе жүктемейміз.

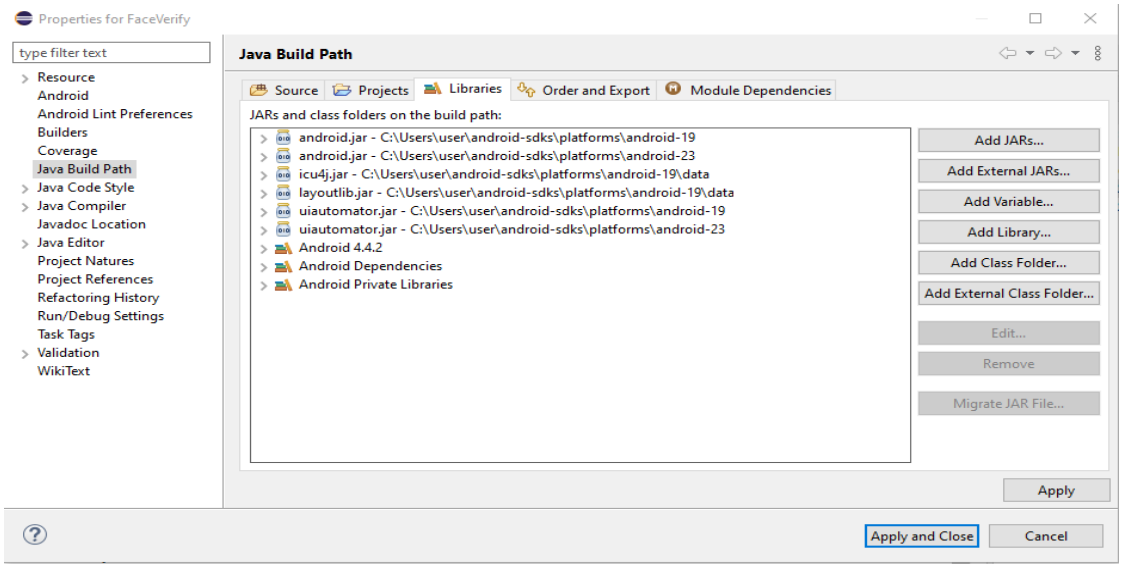
Енді бізді Android нұсқаларының атаулары бар қалталар қызықтырады. Алдымен Android 4.4.2 орнатыңыз. Дамуға ыңғайлы болғаннан кейін сіз әрқашан осында оралып, қажетті Android нұсқаларын жүктей аласыз. Осы уақытта біз дәлелденген Android 4.4.2 (API 19) жұмыс істейміз.

Кем дегенде, әзірлеу үшін бізге Android 4.4.2 (API 19) қалтасында екі компонент қажет) :

- SDK платформасы-әзірлеу үшін қолданылады. Қарапайым сөзбен айтқанда, бұл қосымшаларды құру кезінде қолданылатын Android жүйесінің барлық бағдарламалық компоненттерін қамтиды - мысалы, терезелер, түймелер және т. б.

- ARM Eabi v7a жүйесінің кескіні - Android жүйесінің кескіні. Ол нақты құрылғыларды қоспай-ақ, қосымшаларды тікелей компьютерде сынау үшін қажет болатын Android эмуляторын жасау үшін қолданылады.

Біз Android 4.4.2 нұсқасының компоненттері мен мүмкіндіктерін қолдана отырып Қосымшаны жасай аламыз және осы қосымшаны 4.4.2 нұсқасының эмуляторында іске қоса аламыз. Бұл сізге жұмысты бастау үшін жеткілікті. Осы екі элементті Android 4.4.2 (API 19) қалтасынан тексеріңіз.



3.14-сурет Бағдарламада қолданылатын кітапханалрмен қосымшалар

Егер сізге Android-тің басқа нұсқалары қажет болса, SDK менеджеріне өтіп, қажетті нұсқа үшін осы екі компонентті орнатыңыз.

Төменгі оң жақ бұрышта түймесін басыңыз пакеттерді орнату <нөмір>, біз таңдаған барлық компоненттерді орнату үшін. Платформалардың өлшемі бірнеше жүз мегабайт болуы мүмкін екенін ескеріңіз.

Орнату тізімі бар терезе пайда болады, лицензияны қабылдауды белгілеп, Орнату түймесін басыңыз. Процесс жалғасып, пайыздар, килобайттар мен секундтар төменнен ерекшеленді. Процесс аяқталғаннан

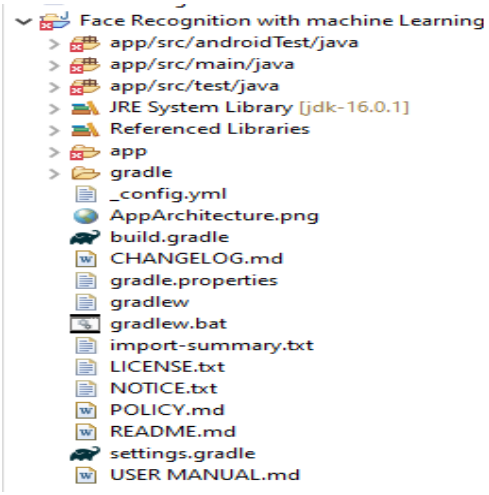
кейін төменгі бөлігінде "пакетті жүктеу"деген жазу жасалады. Барлығы жұмысқа дайын. Біз жабыламыз және Android қосымшаларын қалай жасау керектігін біле бастаймыз.

Даму ортасының барлық параметрлері аяқталғаннан кейін жобаға қажетті ресурстарды қосу керек.

IDE-ді орнату сонымен қатар дамытуға қажетті кітапханаларды орнатуды қамтиды. Конфигурациялау үшін Window / Preferences / Java / Build Path / User Libraries тармағына өтіп, Add External JARs қосу пәрменін таңдаңыз.

Java тілі қосымшаларды тез және ыңғайлы түрде дамыту мүмкіндігі үшін таңдалды. Сонымен қатар, оның тілмен жұмыс істеуін жылдамдатуға арналған көптеген плагиндер мен кеңейтімдер бар. Сонымен қатар, тілде көптеген стандартты кітапханалар бар.

Бағдарламалық жасақтаманы жобалаудың жақсы принципі - қосымшаны құрылымдық бөліктерге бөліп, олардың дамуын жеңілдету. Бұл тезис жобасында бұл әр түрлі технологияларды қолдану түрінде де (Java, Android SDK, OpenCV, CSS), сонымен қатар ресурстарды олардың функционалдығына қарай иерархиялық каталогтарға бөлу түрінде де көрініс табатын болады. Src / it / cv каталогы қосымшаны іске қосуы қажет негізгі файлдарды осылай сақтайды (негізгі кластар, түзету файлдары және т.б.). Src / it / cv / tools каталогында қосымшаның көмекші сыныптары болады. (3.15 суретті қараңыз).



3.15-сурет Жобаның құрылымы

3.3 OpenCV қолдану

OpenCV кітапханасы бейне ағынымен, суреттермен жұмыс істеуге арналған қосымшаның негізгі логикалық негізі болып табылады және осы тезистің негізгі мақсаты - адамның бет-әлпетін анықтау және оны монитор экранында бақылау.

OpenCV және ADT дұрыс жұмыс істеуі үшін кіріс жақтауын камерадан

екі технология үшін де ыңғайлы деректер форматына түрлендіру керек. Бұл процесс көмекші құралдар класында сипатталған.

Бұл технологияның қосымшадағы жұмысы камераны іске қосу батырмасынан басталады, OpenCV кітапханасындағы әдістер бейнелерді бейне ағынынан түсіруге және оларды бірден сұр реңктерге айналдыруға және бет-әлпетті дәл тану үшін кескіннің гистограммасын әртүрлі арналар арқылы туралауға мүмкіндік береді (мысал).

```
// Жақтауды сұр реңктерге түрлендіру
Imgproc.cvtColor(frame, grayFrame, Imgproc.COLOR_BGR2GRAY);
// Нәтижені жақсарту үшін гистограмманы тегістеңіз
Imgproc.equalizeHist(grayFrame, grayFrame);
```

Бетті тану алдын-ала бетті танудың 2 түрлі әдісін қолдану арқылы жүзеге асырылады: Хаарды түрлендіру және жергілікті екілік шаблондар (LBS). Бұл дипломдық жоба анықтаудың екі әдісін де ұсынады. Оларды бағдарламада қолдану үшін алдын-ала дайындалған каскадтар адамның бет-әлпетін анықтау үшін қолданылды.

Сонымен қатар, OpenCV белгіленген адамның бетіне тіктөртбұрыштың суретін салады. Егер кадр кескінінің кем дегенде 20% - ын алатын болса, бет анықталады. Бет өлшемінің шегі азайған кезде, суретте әлсіз жүйелерге теріс әсер етуі мүмкін орталық процессорға жүктеменің бірнеше есе артуы байқалды, осыған байланысты ең төменгі бет кескінінің оңтайлы мөлшері таңдалды.

```
if (this.absoluteFaceSize == 0)
{
int height = grayFrame.rows();
if (Math.round(height * 0.2f) > 0)
{
this.absoluteFaceSize = Math.round(height * 0.2f);
}
}
```

Бетті анықтаудың максималды қашықтығы камерадағы суреттердің сапасы мен мөлшеріне байланысты, сондықтан кіріс бейне ағынының ажыратымдылығы неғұрлым жоғары болса, қолданба тани алатын тұлға соғұрлым аз болады. Осы диссертацияның өмір қауіпсіздігі бөлімінде сипатталған қажетті жағдайларды орындау кезінде адамды дұрыс сәйкестендіру жүзеге асырылады.

Камераны іске қосу және одан әрі өңдеу үшін кіріс бейне ағынын жою бірнеше таңдау процедураларының арқасында жүзеге асырылады.

```
@FXML
protected void startCamera()
{
if (!this.cameraActive)
{
//              Отключение
```

```
чекбоксов
this.haarClassifier.setDisable(
true);
this.lbpClassifier.setDisable(t
rue);
        //      Начало
захвата        видео
this.capture.open(0
);
        // Проверка видеопотока на
доступность                    if
(this.capture.isOpened())
        {
        this.cameraActive = true;
        // Захват кадры каждые 33 мс (30
кадров/сек) Runnable frameGrabber =
new Runnable() { @Override
        public void run()
        {
        // Захват 1 кадра
        Mat frame = grabFrame();
        // Конвертирование и показ кадра
        Image          imageToShow          =
Tools.mat2Image(frame);
updateImageView(originalFrame,
imageToShow);
        }
        };
        this.timer      =      Executors.newSingleThreadScheduledExecutor();
this.timer.scheduleAtFixedRate(frameGrabber,          0,          33,
TimeUnit.MILLISECONDS);
        // Обновление надписи кнопки
запуска                        камеры
this.cameraButton.setText("Stop Camera");
        }
        else
        {
        // Логгирование ошибки
        System.err.println("Failed to open the camera connection...");
        }
        }
        else
        {
        //
```

```
Недоступность
камеры
this.cameraActive =
false;
        // Повторное обновление надписи кнопки
запуска камеры this.cameraButton.setText("Start
Camera");
        //              Включение
чекбоксов
this.haarClassifier.setDisable(f
alse);
this.lbpClassifier.setDisable(fa
lse);
        //
Остановка
таймера
this.stopAcquisit
ion();
        }
        }
```

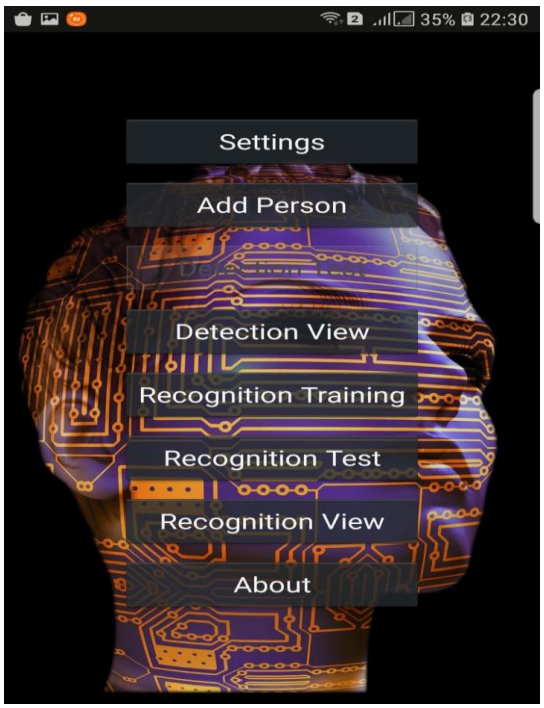
Бет-әлпетті тану ықтимал нақты тұлғалардың массивін құру арқылы жүзеге асырылады.бастапқыда бейне ағыны жеке кадрларға бөлінеді және анықтау сапасын жақсарту үшін әр кадр қара-ақ түске боялады. Осыдан кейін әр кадр кадр биіктігінің 20% көлеміндегі жеке квадраттарға бөлінеді және әр шаршы қолданыстағы Хаара каскадына немесе LBSH-ге сәйкес келеді. Егер нәтиже оң болса, онда осы шаршы туралы ақпарат бет массивіне енгізіледі және болжамды шаршының айналасында шекараны анықтайтын шекаралар салынады. Бұл процесс нақты уақытта жүреді және кідіріс адамның көзіне көрінбейді, оған бұл адам бақыланып жатқан сияқты. Кодтың қысқартылған нұсқасы төменде берілген.

```
        private void detectAndDisplay(Mat frame)
        {
        MatOfRect faces = new
MatOfRect(); Mat grayFrame =
new Mat();
        // Определение лиц
        this.faceCascade.detectMultiScale(grayFrame, faces, 1.1, 2, 0 |
Objdetect.CAS- CADE_SCALE_IMAGE,
        new Size(this.absoluteFaceSize, this.absoluteFaceSize), new Size());
        // Выбор цвета и прорисовка квадратов вокруг лиц
        Scalar color = new
      Scalar(recColor.getValue().getBlue()*255,recColor.get-
Value().getGreen()*255,recColor.getValue().getRed()*255);
        Rect[] facesArray = faces.toArray();
```

```
for (int i = 0; i < facesArray.length; i++)
Imgproc.rectangle(frame, facesArray[i].tl(), facesArray[i].br(), color, 2);}
```

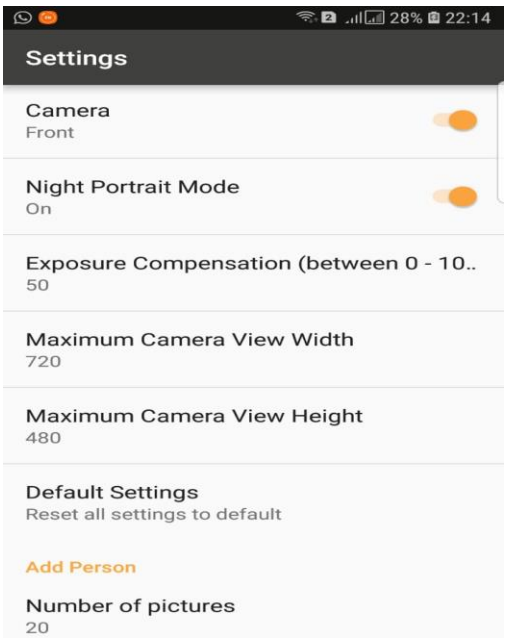
3.4 Жобаны жүзеге асыру және тестілеу

Бағдарламаның негізгі терезесі бірнеше батырмалардан тұрады. Олар: детектор параметрлерін таңдауға арналған мәзірден, деректер қорын қолдан толтыруға арналған парақшаның батырмасы сонымен қатар бетті тануды бастауға арналған белсенді түймеден тұрады.

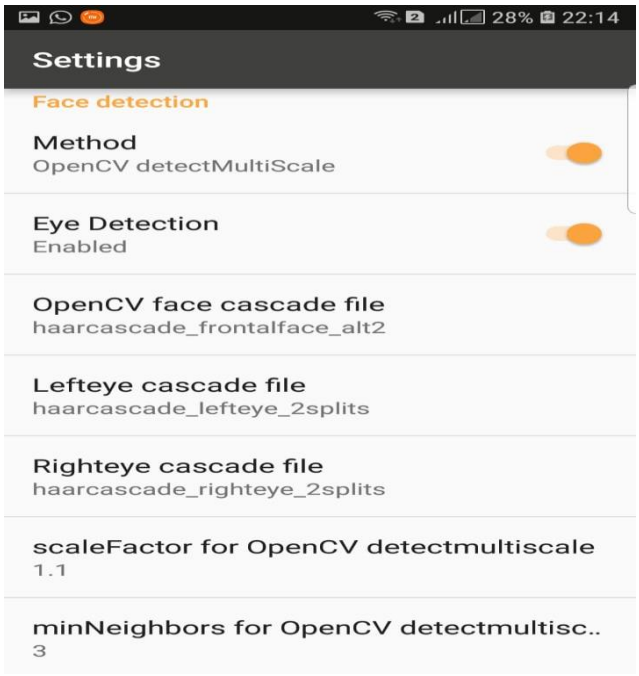


3.4.1-сурет Бағдарламаның басты беті

Мобильді қосымшаның баптаулары терезесінде, камераның бартаулары мен копьютерлік көру технологиясын реттеуге болады.

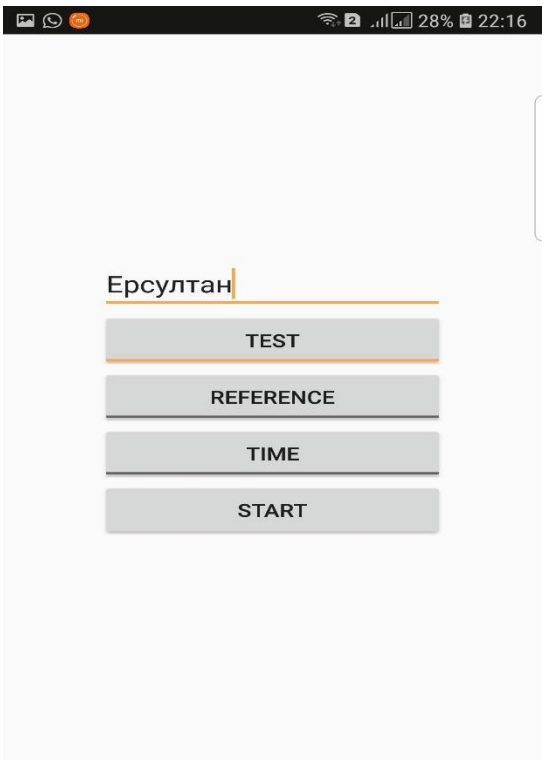


3.4.2-сурет Баптаулар терезесі

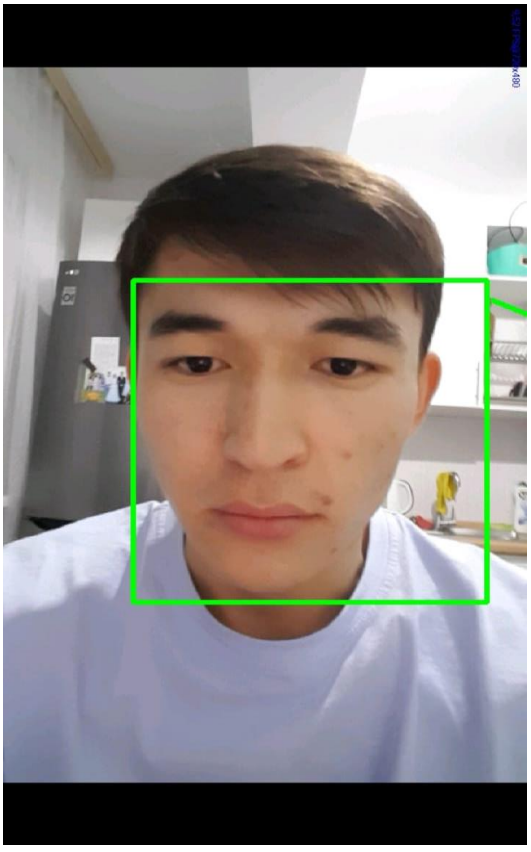


3.4.3-сурет Баптаулар терезесі

Мобильді қосымшада мәліметтер базасына қолдан мәліметтерді енгізіге болады. Мәліметтер базасында 1 турға үшін әр түрлі ракурста 20 сурет енгізу керек.



3.4.4-сурет Мәліметтер енгізу терезесі



3.4.5-сурет Мәліметтер базасына тұлғаның суретін қою терезесі

Мобильді қосымшада тұлғаның мәліметтер қорындағы сәйкестігін

тексеру процесі кезінде бір уақытта бірнеше тұлғаны тану мүмкіндігі бар.

3.4.6-сурет Тұлғаны тану терезесі

Қорыта келе бұл бағдарламаны Андроид операциялық жүесімен жұмыс жасайтын барлық құрылғыларда қолдануға болады. Бағдарлама дәл ұақыт үстінде бірнеше тұлғаға дейін тани алады.

4 Өмір-тіршілік қауіпсіздігі

4.1.1 Еңбек процесінде, әзірлеушіге әсеретететін жұмыс орны мен факторлардың сипаттамалары. Жұмыс орны.

Жұмыс орны – бұл адам-оператор мәселесін сәтті шешуге қолайлы жағдайларды қамтамасыз ететін Функционалды және кеңістіктік ұйымдастырылған техникалыққұралдар мен еңбек объектілерінің жүйесі. Адам өмірінің қауіпсіздігі мәселелері бағдарламаны әзірлеу, іске асыру немесе пайдалану болсын, өмірлік циклдің барлық кезеңдерінде шешілуі керек.

Адамның қауіпсіз өмірін қамтамасыз ету көбінесе қауіпті, зиянды өндірістік факторларды дұрыс бағалауға байланысты. Адам ағзасындағы ауырлық дәрежесінің өзгеруі әртүрлі себептерге байланысты болуы мүмкін. Бұл жұмыс ортасының кез-келген факторлары, шамадан тыс физикалық және психикалық стресс, нейро-эмоционалды стресс, сондай-ақ осы себептердің әртүрлі комбинациясы болуы мүмкін.

Осы тарауда мен дайын өнімді ақаулардың бар-жоғын бақылауға, олардың спектрлік графиктерін зерттеу арқылы жұмыс істейтін жабдықтағы ақауларды диагностикалауға және анықтауға арналған бағдарламалық жасақтаманы әзірлеу сатысында қауіпсіз өмір мәселелерін шешемін.

Ұтымды ұйымдастырылған жұмыс орны еңбек өнімділігін 8-20% арттырып, компьютердің денсаулыққа зиянды әсерін азайтады.

Жұмыс орнын ұйымдастыру әдістерішешілетін міндеттердің сипатына, қолданылатын жабдыққа, адамның нақты жұмысына байланысты.

1 осы жүйені жасау кезінде бағдарламашыға әсер ететін қауіпті және зиянды факторларды талдау.

Қауіпті және зиянды өндірістік факторлар табиғаты бойынша келесі топтарға бөлінеді:

- Жеке;
- Химиялық;
- Психофизиологиялық;
- Биологиялық.

Зертханада бағдарламашыға келесі физикалық факторлар теріс әсер етуі мүмкін:

- жоғары және төмен ауа температурасы;
- Ауадағы шаң мен газдың шамадан тыс мөлшері;
- Жоғары және төмен ылғалдылық;
- Жұмыс орнының жеткіліксіз жарықтандырылуы;
- рұқсат етілген нормадан асатын Шу;
- Иондаушы сәулеленудің жоғары деңгейі;
- Электромагниттік өрістердің жоғары деңгейі;
- Статикалық электрдің жоғары деңгейі;
- Электр тогының соғу қаупі;

- Дисплей экраны күңгірт.

Бағдарламалаушыға тұрақты әсер ететін химиялық қауіпті факторларға мыналар жатады:

- Компьютер жұмыс істеп тұрған кезде ауаның иондалуы нәтижесінде белсенді бөлшектердің пайда болуы.

Бұл бөлмеде биологиялық зиянды өндірістік факторлар жоқ.

Жұмыс ауысымы кезінде операторға әсер ететін психологиялық зиянды факторларға мыналар жатады:

- жүйке-эмоциялық жүктеме;
- психикалық күйзеліс;
- көру анализаторының шамадан тыс кернеуі.

Әрі қарай, осы жүйенің дамуына байланысты пайда болатын бағдарламашыға әсер ететін қауіпті және зиянды факторлар толығырақ қарастырылады.

4.1.2 ДК - мен жұмыс істеу кезінде дұрыс отыру

Әзірлеуші жұмыс уақытының көп бөлігін отыруға жұмсайды. Бұл позиция энергияны тұтынуды азайтуға, жұмыс кезінде шаршауды азайтуға мүмкіндік береді.

Бұл ұстаным кейбір аспектілерді ескере отырып, ұзақ мерзімді жұмыс үшін ең оңтайлы болып табылады:

-Артқасы бірнеше градусқа бүгілген (бұл позиция омыртқаны түсіруге мүмкіндік береді);

-Қол кресло шынтақшасына еркін түсірілу керек;

-Шынтақ және білек буындары босаңсыған, қолдардың білектері бар жалпы осі бар: олар бүгілмейді немесе иілмейді;

- аяқтар еденге немесе тірекке мықтап тұру қажет.

4.1.3 Әзірлеушінің жұмыс орнының құрамы

Бағдарламашының өндірістік қызметі оны ұзақ уақыт отыруға мәжбүр етеді, бұл мәжбүрлі қалып болып табылады, сондықтан денеге үнемі ұтқырлық пен күшті физикалық белсенділік жетіспейді. Отырған кезде иық белдеуі маңызды рөл атқарады. Кеңістіктегі қолдың қозғалысы иық белдеуі мен артқы бұлшықеттердің жұмысына ғана емес, сонымен қатар омыртқаның, жамбастың және тіпті аяқтың жағдайына да әсер етеді.

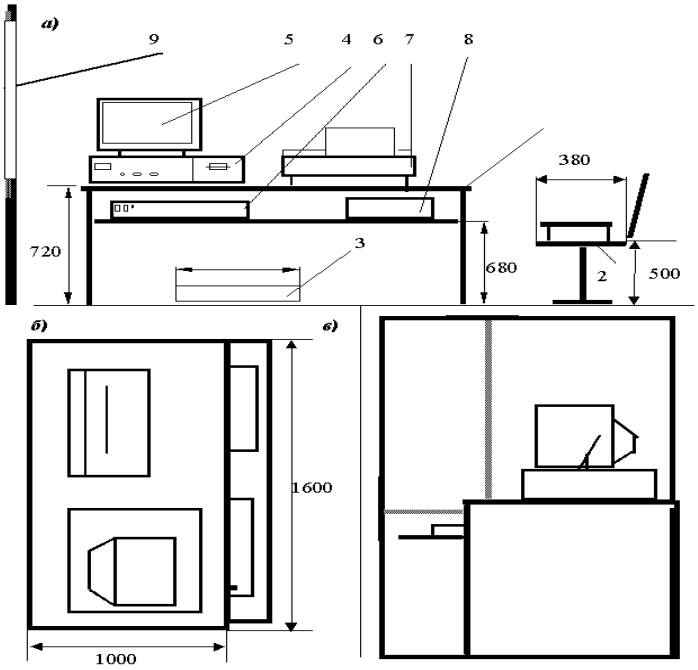
Аурулардың пайда болуын болдырмау үшін позаларды еркін өзгерте білу керек. Негізгі жұмыс қалпын сақтауға кірмейтін тірек-қимыл аппаратының буындарына "алаңдататын" бұлшықет жүктемелерімен толтырылған үзілістермен жұмыс және демалыс режимін сақтау қажет.

Адамның антропологиялық сипаттамалары оның жұмыс орнының жалпы және жоспарлау параметрлерін, сондай-ақ оның жеке элементтерінің бос параметрлерін анықтайды.

Жұмыс жағдайына сәйкес, бағдарламашының жұмыс орны отыруға арналған жеке жұмыс орнына жатады.

Бағдарламашының жұмыс орны кемінде 6 м2, бөлменің биіктігі кемінде 4 м, ал көлемі бір адамға кемінде 20 м3 болуы керек. Зертханада бағдарламашының жұмыс орнын талдағаннан кейін, бұл жұмыс орнының ауданы 4 м2, ал көлемі 12 м3 екендігі анықталды, бұл жоғарыда көрсетілген талаптарға сәйкес келмейді. Сондай-ақ, талдау нәтижесінде бағдарламашының жұмыс орнын ұйымдастырудағы бұзушылықтар анықталды. Осыған байланысты мен бағдарламашының жұмыс орнын келесідей ұйымдастыруды ұсынамын. Оператор жұмыс істейтін жұмыс бетінің еденінен жоғары биіктігі 720 мм болуы керек.қажет болған жағдайда оператордың жұмыс үстелін 680-780 мм биіктікте реттеуге болады. үстел бетінің оңтайлы өлшемдері 1600 х 1000 шаршы метрді құрайды. ММ. үстелдің астында 650 мм тереңдікте аяқтар үшін орын болуы керек. оператордың жұмыс үстелінде үстелдің бетіне 15 ° бұрышта орналасқан аяқ тірегі болуы керек. Тіреудің ұзындығы 400 мм, ені 350 мм. Пернетақтаның үстелдің шетінен қашықтығы 300 мм-ден аспауы керек, бұл операторға білектерге ыңғайлы қолдау көрсетеді. Оператордың көздері мен бейне дисплейдің экраны арасындағы қашықтық 40-80 см болуы керек.

Бағдарламашының жұмыс орындығы көтеру және бұрылу механизмімен жабдықталуы керек. Орындықтың биіктігі 400-ден 500 мм-ге дейінгі диапазонда реттелуі керек, орындық тереңдігі кемінде 380 мм және ені кемінде 400 мм болуы керек. Арқаның тірек бетінің биіктігі кемінде 300 мм, ені кемінде 380 мм.орындық арқасының орындықтың жазықтығына көлбеу бұрышы 90-110 °аралығында өзгеруі керек. Оператордың жұмыс орнының схемасы суретте көрсетілген.



4.1-сурет-бағдарламашының жұмыс орнының сызбасы: а) алдыңғы

көрініс; б) жоғарыдан көрініс; в) бүйір көрінісі.
Шартты белгілер: 1) кесте; 2) орындық; 3) аяққа арналған тұғыр; 4) жүйелік блок; 5) монитор; 6) пернетақта; 7) принтер; 8) қағаз науасы; 9) терезе.

4.1.4 Бөлме ішіндегі жұмыс орнының орналасуы

Көбінесе ДК-мен жұмыс істейтін қызметкерлердің жұмыс орындары терезелерден алыс орналасқан және терезе саңылаулары бүйірде орналасқан. Егер дисплей экраны терезенің ашылуына қараса, қорғаныс экрандары қажет. Терезелерді жарық шашатын перделермен, реттелетін перделермен немесе металлдалған күн қорғанысымен жабдықтау ұсынылады.

Монитор, құжаттар, клавиатура олардың беттерінің жарықтығындағы айырмашылық олардың жарық көздеріне қатысты орналасуына қарай 1: 10-дан аспайтындай етіп, ұсынылған 1: 3 мәні кезінде орналастырылуы тиіс. Экрандағы кескіннің жарықтығы 50-100 кд / м2 (номиналды мәні) болса, құжаттың жарықтандырылуы 300-500 люкс болуы керек. Экран әйнегінің жарқыраған жарықтығын, жарқырауын және шағылысуын болдырмау керек.

Дисплей экрандарын тікелей жарық ағынымен жарықтандыруды болдырмау үшін жалпы жарықтандыру шамдары жұмыс орнының бүйіріне, оператордың шолу сызығына және терезелері бар қабырғаға параллель орналастырылады.

Дисплей құрылғыларының операторларының оңтайлы жұмыс жағдайларын қамтамасыз ету үшін үй - жайларды белгілі бір әрлеу қажет: төбеге 0,7 - 0,8; қабырғаларға - 0,5 - 0,6; еденге - 0,3-0,5 шағылысу коэффициенті бар диффузды шағылысатын материалдарды пайдалану керек.

Компьютерлер қолданылатын бөлмелердегі еденнің беті тегіс, шұңқырсыз, сырғып кетпейтін, оңай жуылатын және дымқыл тазалау үшін болуы керек, сонымен қатар антистатикалық қасиеттерге ие болуы керек.

Жұмыс орындарын жоспарлау мониторлары бар жұмыс үстелдерінің арасындағы қашықтықты (бір монитордың және басқа монитордың экранының артқы беті бағытында) ескеруі тиіс, ол кемінде 2,0 м құрауы тиіс, ал бейнемониторлардың бүйір беттері арасындағы қашықтық кемінде 1,2 м болуы тиіс.

4.1.5 Ғимараттағы жұмыс аймағындағы микроклиматқа қойылатын талаптар. Жұмыс санаты

Компьютермен жұмыс жүйелі физикалық белсенділікті қажет етпейтін жеңіл физикалық жұмыс санатына жатады (1а және 16 санаттары). СанПиН 2.2.2.542 - 96 талаптарына сәйкес ДК жұмысы негізгі болып табылатын бөлмелерде 5.1-кестеде келтірілген микроклимат көрсеткіштерінің оңтайлы мәндері қамтамасыз етілуі тиіс.

1-кесте ДК бар үй-ж жұмыс аймағындағы микроклимат көрсеткіштерінің оңтайлы нормалары

Жыл кезеңі	Жұмыс категориясы	Ауа температурасы, ° С	Салыстырмалы түрде ауа ылғалдылығы, %	Ауа жылдамдығы, м / с
Салқын	жеңіл 1а	22-24	40-60	0,1
	жеңіл 1б	21-23	40-60	0,1
Жылы	жеңіл 1а	23-25	40-60	0,1
	жеңіл 1б	22-24	40-60	0,2

Бөлмеге кіретін ауа кем дегенде 19 ° С болуы керек және шаң мен микроорганизмдер болмауы керек. Оңтайлы микроклимат параметрлерін стандартты шектерде және бөлмедегі ауаның қажетті тазалығында автоматты түрде ұстауды қамтамасыз ететін ең тиімді шара-кондиционерді пайдалану.

Егер бөлмедегі ылғалдылықты арттыру қажет болса, күн сайын тазартылған немесе қайнатылған сумен толтырылған ылғалдандырғыштарды қолданыңыз. ДК бар үй-жайдағы ауадағы оң және теріс ауа иондарының деңгейі 2-кестеде келтірілген стандарттарға сәйкес келуі тиіс.

2-кесте ДК-де жұмыс істеген кезде ғимараттағы ауаның иондану деңгейлері

Деңгейлер	1 $см^3$ ауадағы иондардың саны	
	П+	П-
Минималды талап етілетін	400	600
Оңтайлы	1500-3000	30000-50000
Максималды рұқсат етілетін	50000	50000

Ғимараттағы ауа сапасын жақсарту және ғимараттағыауа иондарының қажетті құрамын қамтамасыз ету үшін ДК-мен ғимаратты әр сағат сайын желдету қажет. Ғимараттағы ауаның иондық режимі бұзылған жағдайда ауа ионизаторларын пайдалану керек.

4.2 Жұмыс орнын жарықтандыру

Әзірлеушінің жоғары өнімді жұмысының маңызды шарты оның жұмыс орнын ұтымды жарықтандыру болып табылады. Бұл визуалды аппарат арқылы адам қоршаған әлемнен шамамен 90% ақпарат алады, оны алу көбінесе жарықтандыру сапасына байланысты.

Әзірлеушінің кәсіби қызметі үшін қолайсыз факторлар, көру қабілетіне теріс әсер етеді:

- Жарық деңгейінің төмендеуі, бұл көздің кернеуіне және нәтижесінде тез шаршауға әкеледі.

-Шамадан тыс жоғары жарық шаршауды тудырады, бұл көздің тітіркенуіне және ауырсынуына әкеледі;

Жарықтың дұрыс емес бағыты көзді айтарлықтай шаршататын өткір көлеңкелер немесе күшті шағылысулар жасайды.

ДК-мен үй-ғимаратты және жұмыс орындарын жарықтандыру үшін табиғи, жасанды және аралас жарықтандыруды пайдалану керек. Табиғи жарықтандыру негізінен солтүстікке және солтүстік-шығысқа бағдарланған Жарық ойықтары арқылы қамтамасыз етілуі және табиғи жарықтандыру коэффициентін (КЕО) тұрақты қар жамылғысы бар аудандарда кемінде 1,2% және қалған аумақтарда кемінде 1,5% қамтамасыз етуі тиіс. ДК-мен жарық саңылауларына қатысты жұмыс орындары табиғи жарық бүйірден, негізінен сол жақтан түсетін етіп орналасуы керек. ДК пайдаланылатын үй-жайлардағы жасанды жарықтандыру біркелкі жарықтандырудың жалпы жүйесінің көмегімен жүзеге асырылуы тиіс.

Бөлмеге жақсы енетін табиғи жарық адам психикасына жағымды әсер етеді, жағымды эмоциялар тудырады, жақсы гигиеналық еңбек жағдайларын қамтамасыз етеді. Табиғи жарықтандырудың арқасында метаболизм, қан айналымы, тыныс алу және орталық жүйке жүйесінің қызметі ынталандырылады, бұл өз кезегінде жоғары еңбек өнімділігін қамтамасыз етеді.

4.2.1 Жарықтандыру жүйесін анықтау

Жарықтандыру жүйесін таңдағанда, біріктірілген жарықтандыру жүйесі тиімдірек екенін ескеру керек, бірақ жалпы жарықтандыру жүйесі гигиеналық болып табылады, өйткені ол жұмыс беттерін жарықтандырудың біркелкілігін қамтамасыз етеді. Жергілікті жалпы жарықтандыруды пайдалану-бұл үлкен инвестициясыз жұмыс орындарында жарықтандырудың жоғары деңгейіне жетудің ең оңай жолы. Жекелеген жұмыс орындарын жарықтандыруға қойылатын жоғары талаптар кезінде жарықтандырудың аралас жүйесі пайдаланылады.

Әр түрлі үй - жайларды жасанды жарықтандыру электр жарық көздерін-қыздыру шамдарын және люминесцентті лампаларды, люминесцентті лампаларды қолдану арқылы жүзеге асырылады. Люминесцентті лампалар бірқатар себептерге байланысты өндірістік бөлмелерді жарықтандыру үшін кеңінен қолданылады.

- Шамдардың осы түрін пайдалану қыздыру шамдарымен (пәк) салыстырғанда электр энергиясын бірдей тұтынған кезде 1,5 - 2 есе көп жарық алуға мүмкіндік береді);

- Люминесцентті шамның жарығы жұмсақ, спектрінде табиғи күндізгі жарыққа жақын, көлеңкелердің мүлдем болмауы, бұл түстер мен реңктерді жақсырақ ажыратуға мүмкіндік береді;

- Сонымен қатар, флуоресцентті лампалар қыздыру лампасының қызмет ету мерзімінен 10-12 есе ұзақ қызмет етеді.

Шамдардың бұл түрінің кемшіліктері:

- Жоғары орнату құны;
- Жарық ағынының қоршаған орта температурасына тәуелділігі;
- Жарық ағынының айтарлықтай пульсациясы.

4. 2.2 Жасанды жарықтандыруды есептеу

Әзірлеушінің жұмыс орны орналасқан ғимарат мынадай сипаттамаларға ие:

- бөлменің ұзындығы-6 метр;
- бөлменің ені-5 метр;
- бөлменің биіктігі-3 метр;
- терезелер саны - 1;
- жұмысорындарының саны-3;
- жарықтандыру: табиғи (бүйірліктерезеарқылы) және жалпы жасанды.

Коэффициенттер әдісін қолдана отырып, шамдардың қажетті санын табыңыз.

Жалпы жарықтандыру жүйесі коэффициент әдісін қолдана отырып есептеледі жарықтың қатынасы арқылы көрінетін жарық ағынын қолдану

барлық шамдардың жалпы ағынына есептік бетке түсетін ағын.

Оның мәні шамның сипаттамаларына, бөлменің көлеміне байланысты,

қабырғалар мен төбелердің түстері, қабырғалардың шағылысуымен және

төбеге.

Әр Шамдағы шамның қажетті жарық ағыны мына формула бойынша есептеледі:

$$F_0 = \frac{\mathrm{E}*\mathrm{k}*\mathrm{S}*\mathrm{Z}}{\eta*N}, \qquad (4.1)$$

мұндағы Е-берілген минималды жарық, lx (E = 500);

k-жарық ағынының азаюын ескеретін беріктік қорының коэффициенті

пайдалану кезінде шамдардың ластануы нәтижесінде шамдар

(люминий. шамдар-1,5);

S-жарықтандырылған алаң, $м^2$ (S = 30);

Z - орташа жарықтың минимумға қатынасы (әдетте

люминалар үшін 1,1-1,2 тең деп қабылданады. шамдар - 1.1);

Ldr шамының түрі (2х40 Вт). Ұзындығы 1,24 м, ені 0,27 м, биіктігі 0,1 м.

L - көршілес шамдар арасындағы қашықтық (люминесцентті шамдар қатарлары), La (бөлменің ұзындығы бойынша) = 1,76 м, LV (бөлменің ені бойынша) = 3 м.

l - шеткі шырақтардан немесе шырақтар қатарынан қабырғаға дейінгі арақашықтық, l = 0,3-0,5 л.

l_a = 0,5 L_a, $l_в$= 0,3 $L_в$

l_a = 0,88 м, $l_в$= = 0,73 м.

ŋ-бірлік үлестеріндегі жарық ағынын пайдалану коэффициенті

(қатынасы жарық ағынының құлайтын, есептік беті, к

барлық шамдардың жалпы ағыны).

Ɒ пайдалану коэффициенті шамның түріне, коэффициенттерге байланысты

$ρ_п$ төбесінің, $ρ_с$ қабырғаларының, $ρ_р$ жобалық бетінің, индекстің көрінісі

формула бойынша есептелетін Үй-жайлар:

$$i = \frac{S}{h*(a+b)}, \quad (4.2)$$

мұндағы h-шамның жұмыс бетінен биіктігі;

a-бөлменің ұзындығы;

b-бөлменің ені.

Формула бойынша h табыңыз:

h = H - hp- hc= 3 - 0,7 – 0,25 = 2,05 (м)

мұндағы H-үй-жайдың биіктігі, м (H = 3);

h_pеденнен жұмыс бетінің биіктігі, м (h_p = 0,7);

h_c-негізгі төбеден шамның биіктігі, м h_c= 0,25).

$$i = \frac{6*5}{2,05*(6+5)} = \frac{30}{22,55} = 1,33$$

Ашық фон үшін мынаны алайық: $ρ_п$= 70, $ρ_с$ = 50, $ρ_р$ = 10 =>ŋ=30% .

$$F_л = \frac{F_0}{N}, \quad (4.3)$$

мұндағы $F_л$-бір шамның жарқын ағыны;

F_0-жалпы жарық ағыны;

N-шамдар саны;

ŋ-жарық ағынын пайдалану коэффициенті.

$$F_0 = \frac{300 * 1,3 * 30 * 1,1}{0,3 * 2} = 21450 лм$$

Біз шамдардың санын көрсетілген формула бойынша есептейміз:

$$N = \frac{F_0}{F_л}, \quad (4.3)$$

Жарықтандыру үшін біз ЛХБ65,, жарық сияқты флуоресцентті лампаларды таңдаймыз

олардың ағымы$F_л$ = 4400 Лм құрайды.

$$N = \frac{21450}{4400} = 4{,}8 => 5$$

Шамдардың саны жарықтандыру мөлшеріне байланысты таңдалады

Үй-жайлар, бұл ретте шамдардың саны мынандай болуы тиіс:

олардың арасындағы қашықтықтың олардың жер үстінен іліну биіктігіне қатынасы бұл 1,5÷ 2-ге тең болды.

Жарықтандыру құрылғыларын таңдағанда біз типті шамдарды қолданамыз

ЛСПО 2.. Әр шам екі шаммен жабдықталған. Орналастырылғаншамдар үш қатарда, әр қатарда екеу.

Таңдалған шамның жарық ағынының рұқсат етілген ауытқуы (ε)

есеп айырысу-10% - дан + 20% - ға дейін.

$$\mathrm{E}_{\text{факт}} = \frac{\Phi_{л} * N * N_{л,св} * \eta}{S * z * k}, \quad (4.4)$$

$$\mathrm{E}_{\text{факт}} = \frac{4400 * 5 * 2 * 0{,}36}{30 * 1{,}1 * 1{,}3} = 923\text{лк}$$

Стандартталған деңгейден айырмашылығы:

$$\frac{\mathrm{E}_{\text{факт}} - \mathrm{E}_{\text{норм}}}{\mathrm{E}_{\text{норм}}} * 100\%$$

$$\frac{923 - 500}{500} * 100\% = 84.6$$

Дисплейлердің экрандарын тікелей жарықпен жарықтандыруды болдырмау үшін ағындармен жалпы жарықтандыру шырақтары жұмысшы жағынан орналастырылады, орындар оператордың көру сызығына параллельжәнетерезелерібарқабырғада. Мұндай шамдарды орналастыру олардың дәйекті болуына мүмкіндік береді және табиғи жарық мөлшеріне байланысты қосу және жарықпен көлеңкенің ауыспалы жолақтарымен көздің тітіркенуін жояды, шамдардың көлденең орналасуынан пайда болады.

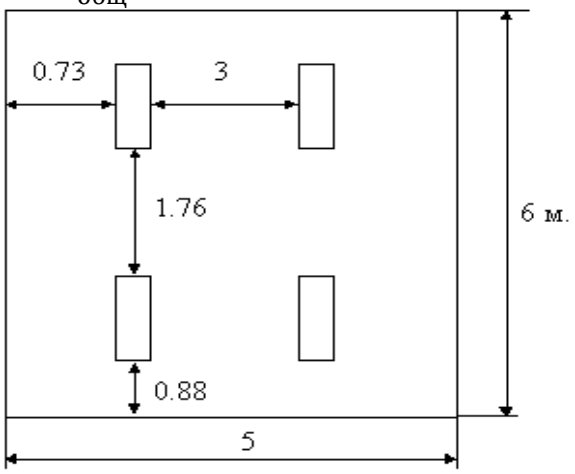
Бүкіл жарықтандыру жүйесінің электр қуаты мына формула бойынша есептеледі:

$$P_{\text{общ}} = P_1 * N, \quad (4.5)$$

Мұнда P_1-1 шамның қуаты=65Вт;

N- шамдар саны=2.

$$P_{\text{общ}} = 65 * 2 = 130\text{Вт}$$



4.2- сурет шамның орналасуы.

5. Техникалық-экономикалық негіздеме

Дипломдық жобаны жасау кезінде тулғаны тануға арналған мобильді қосымша құру керек. Бұл жобаның мақсаты – бағдарламалық өнімді жасап шығару, дәлірек: машиналық оқытуды пайдаланатын тулғаны тануға арналған мобильді қосымшаны құру. Қазіргі әлемде көптеген міндеттер бағдарламалық жасақтаманың көмегімен шешіледі.

Бағдарламалық өнімді әзірлеу кезінде шартты және шартсыз өтулердің болмауы, операторларды бір жолға жазу (шамалы ерекшеліктерді қоспағанда), бағдарламалаудың сызықтық тәсілі сияқты құрылымдалған бағдарламалау бойынша қазіргі заманғы ұсыныстарды ескеру қажет. Сондықтан бағдарламадағы жолдар санын бағдарламадағы операторлар санына жатқызуға болады.

5.1 Бағдарламалық жасақтаманы әзірлеудің көлемі мен күрделілігін анықтау

Бағдарламалық жасақтаманы әзірлеу шығындарының жоспарлы сметасын есептеу үшін “Дипломдық жумыстың экономикалық бөлімін іске асыруға арналған әдістемелік нұсқаулығын” қолдана отырып коефицеттер мен мәндерін аламыз. Бағдарламалық өнімнің жалпы көлемі (V_0) бағдарлама іске асыратын функциялардың саны мен көлеміне қарай айқындалады:

$$V_0 = \sum_{i=1}^{n} \mathrm{V}_i, \qquad (5.1)$$

мұндағы V_i -жеке бағдарламалық функцияның көлемі;

n-функциялардың жалпы саны.

Бағдарламалық жасақтама өнімінің көлемін бағалау өнімнің өлшемі үшін ең қолайлы өлшем бірлігін таңдауды қамтиды. Іс жүзінде келесі өлшем бірліктері кеңінен қолданылады:

-Бастапқы код жолдарының санын анықтау (CDE, LOC жолдары);

-Функция нүктелерін анықтау (функция нүктесі, FP);

-Сипат нүктесінің мәні (сипат нүктесі, PP);

-Деректер ағынының диаграммасындағы "көпіршіктердің" санын анықтау (мәліметтер ағынының диаграммасы, DFD);

- Нысан диаграммасындағы нысандар саны (нысан қатынастарының диаграммасы, ERD); - нысан диаграммасындағы нысандар саны (нысан қатынастарының диаграммасы, ERD);

- Процесс / басқаруға сәйкес келетін "квадраттар" саны (PSPEC / CPEC);

- Басқару сипаттамасындағы әртүрлі элементтердің саны (элемент);

- Құжаттама көлемі (жолдар саны, жолдар саны);

- Объектілер диаграммасындағы объектілер, атрибуттар және қызметтер саны (субъектілер, атрибуттар, қызметтер).

Дипломдық жобада бастапқы код жолы (LOC) бағдарламалық жасақтама көлемін өлшеу бірлігі ретінде қолданылады. LOC есептеу кезінде келесі ұсыныстарды орындау керек:

- Деректердің анықтамасын тек бір рет қарастыру;
- Түсініктемелері бар жолдарды елемеу;

Жөндеу кодын немесе басқа уақытша кодты (сынақ бағдарламалық жасақтамасы, тестілеу құралдары, әзірлеу және прототиптеу құралдары және басқа құралдар) ескермеңіз);

- Қайта пайдалануға болатын бастапқы код мәлімдемелерін елемеңіз.

Бағдарламалық өнім көлемін (бастапқы код жолдарының санын) есептеу бағдарламалық жасақтама түрін анықтауды, бағдарламалық жасақтама функцияларын кешенді техникалық негіздеуді және әр функцияның көлемін анықтауды қамтиды. Жобаның техникалық-экономикалық негіздемесі кезеңінде бұрын аяқталған ұқсас жобалар бойынша қолда бар нақты деректер негізінде немесе қолданыстағы стандарттарды қолдану жолымен ғана шамамен алынған бағалар алынуы мүмкін. Әзірленетін бағдарламалық қамтылымның функциялары туралы ақпарат негізінде функциялар каталогына сәйкес функциялар көлемі және бағдарламалық қамтылымның жалпы көлемі айқындалады, ол ұйымда бағдарламалық қамтылымды әзірлеу шарттарын ескере отырып нақтыланады (түзетіледі). Жаңартылған бағдарламалық қамтылымның көлемі (Vy) мына формула бойынша есептеледі:

$$V_y = \sum_{i=1}^{n} V_{yi}, \qquad (5.2)$$

мұндағы V_{yi} -жеке бағдарламалық функцияның (LOS) берілген көлемі.

Бағдарламалық қамтамасыз етудің қайта қаралған көлемі мен көлем бірлігіне жұмсалатын еңбек шығындарының негізінде бағдарламалық қамтамасыз етуді әзірлеудің стандартты және жалпы еңбек сыйымдылығы айқындалады. Бағдарламалық қамтамасыз етуді әзірлеудің нормативтік күрделілігі (Тн) есепке қабылданған көлем (Vy) және күрделілік санаты (В қосымшасы) негізінде айқындалады, ол жобаның күрделілігі мен жаңалығын және әзірлеу кезінде стандартты модульдерді пайдалану дәрежесін ескере отырып көрсетіледі.

Еңбек сыйымдылығы нормативі (Тn) жалпы еңбек сыйымдылығын (То) анықтау үшін негіз болады, оны есептеу жобаның көлеміне байланысты әртүрлі тәсілдермен жүзеге асырылады.

Шағын жобалардың жалпы еңбек сыйымдылығы мынадай формула бойынша есептеледі:

$$T_O = T_{\text{Н}} * K_C * K_m * K_{\text{Н}}\ , \qquad (5.3)$$

мұндағы K_c-бағдарламалық жасақтаманың күрделілігін ескеретін коэффициент;

$К_т$-стандартты модульдерді әзірлеу кезінде пайдалану дәрежесін ескеретін түзету коэффициенті;

$К_н$ -бағдарламалық қамтамасыз етудің жаңашылдық дәрежесін ескеретін коэффициент.

Күрделілік коэффициенті мына формула бойынша есептеледі:

$$K_C=1+\sum_{i=0}^{n} K_i, \quad (5.4)$$

мұндағы K_i -нақты сипаттама есебінен бағдарламалық қамтылымның күрделілігінің ұлғаю дәрежесіне сәйкес келетін коэффициент;

Күрделілік коэффициентін есептеу 4.1-кестеде берілген деректерге негізделген.

Барлық бағдарламалық жасақтама әдетте келесі сипаттамалардың болуына (болмауына) байланысты үш күрделілік санатына бөлінеді .

5.1-кесте-Бағдарламалық қамтамасыз етудің күрделілік санаттарының сипаттамасы

Қиындық категориялары	Бағдарламалық жасақтаманың сипаттамалары
1	Мынадай сипаттамалардың біреуіне немесе бірнешеуіне ие бағдарламалық қамтамасыз ету: 1) пайдаланушымен күрделі интеллектуалды тілдік интерфейстің болуы. 2) нақты уақыт режиміндегі жұмыс режимі. 3) деректерді телекоммуникациялық өңдеуді және қашықтағы объектілерді басқаруды қамтамасыз ету. 4) компьютерлік графика. 5) көп машиналы кешендер. 6) есептеулерді Елеулі параллелизациялауды қамтамасыз ету.
2	Мынадай сипаттамалардың біреуіне немесе бірнешеуіне ие бағдарламалық қамтамасыз ету: 1) оңтайландыру есептері.
	2) кіріс және шығыс құрылымын өзгерту үшін бағдарламалық жасақтама конфигурациясын қамтамасыз ету. 3) стандартты емес жабдық конфигурациясы үшін бағдарламалық жасақтаманы баптау. 4) бағдарламалық қамтылымның тасымалдануын қамтамасыз ету болып табылады. 5) Ерекше күрделі инженерлік және ғылыми есептерді орындау.
3	Жоғарыда аталған сипаттамалары жоқ бағдарламалық жасақтама.

Күрделілік факторының күрделілікке әсері Нормативтік еңбек сыйымдылығын тиісті күрделілік коэффициентіне көбейту арқылы ескеріледі.

Күрделілік коэффициенті әзірленетін бағдарламалық өнімнің күрделілігіне байланысты қосымша еңбек шығындарын ескереді (4.1-кестені қараңыз).

Кесте 5.2-Бағдарламалық қамтамасыз етудің қосымша күрделілік факторлары

Бағдарламалық жасақтаманың сипаттамасы	Құндылықтар $К_с$
1.Бағдарламалық жасақтаманың кеңейтілген операциялық о ртадағы жұмысы(басқа бағдарламалық жасақтамамен байланыс)	0,08
2. Интерактивті қол жетімділік	0,06
3. Күрделі құрылымдарда деректерді сақтауды, қызмет көрсетуді және іздеуді қамтамасыз ету	0,07
Бағдарламалық жасақтама бір уақытта бірнеше сипаттамаларға ие 4.1 2 сипаттамалары 4.2 3 сипаттамалары 4.3 3-тен көп сипаттама	 0,12 0,18 0,26

Бағдарламалық қамтамасыз етуді әзірлеу кезінде стандартты модульдерді ($К_т$) пайдалану дәрежесін ескеретін түзету коэффициенті жобаланатын өнімнің жалпы көлеміндегі осы модульдердің үлес салмағымен айқындалады (4.2-кестені қараңыз)

5.2-кесте-үлгілік бағдарламалардың стандартты модульдерін және бағдарламалық қамтамасыз етуді ($К_т$) пайдалануды ескере отырып, түзету коэффициентінің мәні)

Стандартты модульдермен, бағдарламалармен және модельдік бағдарламалармен әзірленген бағдарлама үшін іске асырылатын функцияларды қамту дәрежесі	Мәні, $К_т$
1. 60 % - дан жоғары	0,6
2. 40 % -дан 60 %	0,7
3. 20 % -дан 40 %	0,8
4. 20 %-ға дейін	0,9
5.Әзірленген бағдарламалық жасақтаманың функцияларын орындау үшін пайдаланылмайтын типтік бағдарламалар мен бағдарламалық жасақтама	1,0

Бұл фактордың еңбек сыйымдылығына әсерін анықтау кезінде нормативтік еңбек сыйымдылығын тиісті коэффициентке көбейту арқылы есептеледі.

Әзірленген бағдарламалық қамтамасыз етудің жаңалығын ескере отырып түзету коэффициенті ($К_н$) 4.3-кестеде келтірілген деректер негізінде айқындалады.

5.3-кесте-бағдарламалық қамтылымның жаңалығын ескере отырып түзету коэффициенттері ($К_н$)

Жаңалық санаты	Жаңалық дәрежесі	Қолданылуы		$К_н$ Мәні
		Компьютердің жаңа түріне негізделген	Жаңа ОЖ ортасында	

5.3-кестенің жалғасы

A	Баламасы жоқ түбегейлі жаңа бағдарламалық қамтамасыз ету	+ – + –	+ + – –	1,75 1,6 1,2 1,0
B	Бағдарламалық жасақтаманың белгілі бір параметрлік сериясын жасау болып табылатын бағдарламалық жасақтама	+ – +	+ – –	1,0 0,9 0,8
C	Бұрын игерілген конфигурация түрлеріне арналған бағдарламалық жасақтаманың белгілі бір параметрлік сериясын жасау болып табылатын бағдарламалық жасақтама ДК және ОЖ	–	–	0,7

Әзірленген бағдарламалық жасақтаманың сипаттамаларын қолданыстағы аналогтармен салыстыру сарапшыға оның жаңашылдық дәрежесін анықтауға мүмкіндік береді. Егер аналогтар болмаса, онда Бағдарламалық жасақтамаға А санаты беріледі, В және С санаттарындағы бағдарламалық жасақтаманың жаңашылдық дәрежесі қолданыстағы бағдарламалық өнімдердің дамуына сәйкес келеді. Жаңалық коэффициенттерін белгілеу кезінде бағдарламалық қамтамасыз етудің жаңашылдық дәрежесі және оның жаңа немесе игерілген ДК түрлері үшін, жаңа немесе игерілген ОС үшін мақсаты ескеріледі. Жаңалық факторының еңбек сыйымдылығына әсері еңбек сыйымдылығын тиісті жаңалық коэффициентіне көбейту арқылы ескеріледі.

Тиісінше, осы дипломдық жобада әзірленген қосымшаның бағдарламалық жасақтамасы, код жолдарының саны (LOC) тексеріліп, 1030 құрады, сондықтан күрделіліктің 1-ші санатына сәйкес нормативтік күрделілік 51 құрайды.

Күрделілік коэффициенті 4 формула бойынша есептеледі және

$$K_C = 1 + 0{,}12 = 1{,}12.$$

5.4-кесте-бағдарламалық қамтылымның (Vy) берілген көлеміне және бағдарламалық қамтылымның күрделілік тобына байланысты бағдарламалық қамтылымды әзірлеуге (Тн) арналған ірілендірілген уақыт нормалары (адам / күн)

БҚ көлемі (LOC)	Бағдарламалық қамтамасыз етудің күрделілік санаттары			Бағдарламалық қамтамасыз етудің күрделілік санаттары
	1	2	3	
1	2	3	4	5
200	—	—	21	1

5.4-кестенің жалғасы

300	—	—	23	2
400	—	—	2	3
			5	
500	—	—	27	4
600	—	33	28	5
700	—	36	30	6
800	—	38	32	7
900	—	40	34	8
1000	51	43	36	9
1200	54	45	38	10
1400	57	48	40	11
1600	60	50	42	12
1800	64	54	45	13
2000	68	57	48	14

Стандартты $К_т$ = 0,6 модульдерін пайдалану дәрежесін ескере отырып, түзету коэффициенті, өйткені бос кітапханалар мен еркін бағдарламалық қамтамасыз ету қолданылады.

Бағдарламалық қамтылымның жаңалығын ескеретін түзету коэффициенті В және $К_н$ = 0,7 жаңашылдық санатына жатады.

Барлық қажетті коэффициенттерді біле отырып, жалпы еңбек сыйымдылығы 3 формула бойынша есептеледі:$ТО$ = 51 · 1,12 · 0,6 · 0,7 = 23,99.

Кесте 5.5-жалпы еңбек сыйымдылығын есептеу нәтижелері

Шама атауы	Шартты белгі	Мән
Стандартты еңбек сыйымдылығы	$Т_н$	51
Қиындық коэффициенті	$К_с$	1,12
Стандартты модульдерді қолдану дәрежесін ескере отырып түзету коэффициенті	$К_т$	0,6

5.5-кестенің жалғасы

Бағдарламалық жасақтама үшін түзету коэффициенті	$К_н$	0,7
Жалпы еңбек сыйымдылығы	$Т_О$	23,99

5.2 Орындаушылар саны және бағдарламалық қамтамасыз етуді әзірлеу кезеңі

Жалпы еңбек сыйымдылығына сүйене отырып, әзірлеушілердің жоспарланған саны ($Ч_р$) және жобаны тұтастай іске асыру үшін қажетті жоспарланған мерзімдер ($Т_р$) анықталады. Бұл жағдайда келесі міндеттер шешілуі мүмкін:

- жобаны әзірлеудің белгіленген мерзімдеріне орындаушылар санын есептеу;

- орындаушылардың белгілі бір саны үшін жобаны әзірлеу мерзімдерін анықтау.

Жобаны орындаушылардың саны ($Ч_р$) мынадай формула бойынша есептеледі:

$$Ч = \frac{Т_о}{(Т_Р * Ф_{эф})}, \quad (5.5)$$

мұнда $Ф_{эф}$ -бір қызметкердің жыл (күн) ішіндегі жұмыс уақытының тиімді қоры);

$Т_о$ -жобаны әзірлеудің жалпы еңбек сыйымдылығы (адам-күн);

$Т_р$ -жобаны әзірлеу кезеңі (жылдар).

Жобаны әзірлеу уақыты ($Т_р$) мына формула бойынша анықталады:

$$Т_р = \frac{Т_о}{(Ч_р * Ф_{эф})}, \quad (5.6)$$

Бір қызметкердің тиімді жұмыс уақыты қоры ($Ф_{эф}$) мынадай формула бойынша есептеледі:

$$Ф_{Эф} = Д_Г - Д_П - Д_В - Д_О, \quad (5.7)$$

мұндағы $Д_г$ -жылдағы күндер саны;

$Д_п$ -жылдағы Мерекелер саны;

$Д_в$ -бір жылдағы демалыс күндерінің саны;

$Д_о$ - демалыс күндерінің саны

Бұл дипломдық жобадағы өтінімді 1 адам жасаған, сондықтан Chp = 1. Бір қызметкер үшін тиімді уақыт қоры ($Ф_{Эф}$) 7-формула бойынша есептеледі (демалыс күнтізбесі бойынша 2021 жылғы барлық деректер) және:

$$Ф_{Эф} = 365 - 13 - 104 - 14 = 234 \text{ к.}$$

Жобаны әзірлеу мерзімі 4.6 формула бойынша есептеледі :

$$Т_Р = \frac{23,99}{(1*234)} = 0,10 \text{ лет} \approx 38 \text{ дней.}$$

5.3 Ақпараттық технологияларды әзірлеуге арналған шығындарды есептеу

Ақпараттық технологиялар деп Экономикалық ақпараттық жүйелер (ЭАЖ), бағдарламалық өнімдер (ПП), ақпараттық дерекқорлар

және т. б. түсініледі.

Ақпараттық технологиялар ($С_п$) түріндегі жобалық шешімді әзірлеуге арналған жалпы шығындарды есептеу мынадай формула бойынша жүзеге асырылады:

$$С_П = З_{ФОТ} + З_{соц} + Р_М + Р_{со} + Р_{мв} + Р_{нк} + П_з + Р_{нак}, \quad (5.8)$$

мұндағы $З_{ФОТ}$ -құрылыс салушылардың жалақысының жалпы қоры, теңге;

$З_{соц}$ -әлеуметтік салық шегерімдері, теңге;

$Р_м$ -материалдардың құны, теңге;

$Р_{со}$ -жобалық шешімді әзірлеу үшін қажетті арнайы бағдарламалық қамтамасыз етудің құны, теңге;

$Р_{мв}$ -Жабдықты пайдалануға байланысты шығындар, теңге;

$Р_{нк}$ -ғылыми сапарларға арналған шығыстар, теңге;

$П_з$ -басқа да шығыстар, теңге;

$Р_{нак}$ -үстеме шығыстар, теңге.

Құрылыс салушылардың жалақы қорының мөлшері ($З_{ФОТ}$) мына формула бойынша есептеледі:

$$З_{ФОТ} = З_о + З_д, \quad (5.9)$$

мұндағы $З_о$ -базалық жалақы, теңге;

$З_д$ -қосымша жалақы, теңге.

Нақты бағдарламалық жасақтама үшін орындаушылардың базалық жалақысы мына формула бойынша есептеледі:

$$З_о = \sum_{i=1}^{n} Т_{чi} \cdot Т_ч \cdot Ф_п \cdot К, \quad (5.10)$$

мұндағы n-нақты бір жобаны әзірлеумен айналысатын орындаушылар саны

$Т_{чi}$ -і мердігердің сағаттық тарифтік ставкасы, мың теңге;

$Ф_п$ -і Орындаушының жұмыс уақытының жоспарлы қоры, күндер;

$Т_ч$ -күніне жұмыс сағаттарының саны, сағат;

К -бонус коэффициенті.

Базалық жалақыны есептеудің негізі жалпы еңбек сыйымдылығы, жоспарланған қызметкерлер саны және бағдарламалық жасақтаманы әзірлеудің жоспарланған уақыты болып табылады. Еңбекке ақы төлеу тарифтік санаттар мен тарифтік коэффициенттер көрсетілген Қазақстан Республикасының Бірыңғай тарифтік жоспары (БКО) негізінде жүзеге асырылады. Қазақстан Республикасы экономикасының бюджеттен тыс секторында санаты, лауазымы, білімі, орындалатын жұмыстың күрделілігі мен

практикалық тәжірибесін ескере отырып, тарифтік санаттар бойынша қызметкерлерді бөлу жөніндегі Нұсқаулық бар.

Әрбір Орындаушының айлық тарифтік ставкасы ($Т_м$) 1-санаттағы ағымдағы айлық тарифтік ставканы ($Т_{м1}$) белгіленген тарифтік санатқа сәйкес келетін тарифтік коэффициентке ($Т_к$) көбейту жолымен айқындалады:

$$T_М = T_{М1} \cdot T_К, \quad (5.11)$$

Жалақының сағаттық ставкасы жалақының айлық ставкасын 40 сағаттық апталық жұмыс уақыты кезінде белгіленген сағатпен септелген орташа айлық жұмыс уақытына ($Ф_р$) бөлу жолымен есептеледі:

$$Т_ч = \frac{Т_м}{Ф_р}, \quad (5.12)$$

мұндағы $Т_ч$ -сағаттық тарифтік мөлшерлеме, мың теңге;

$Т_м$ -айлық тарифтік мөлшерлеме, мың теңге.

Қосымша жалақы мына формула бойынша есептеледі:

$$З_д = \frac{З_0 * Н_д}{100\%}, \quad (5.13)$$

мұндағы $Н_д$ -қызметкерлердің қосымша жалақысының нормативі,%. Әлеуметтік салық жұмыскердің табысынан 11% - ды құрайды (ҚР СК 358-Б., 1-т.) және мына формула бойынша есептеледі:

$$З_{соц} = (З_{ФОТ} - ПО) \cdot 11\%, \quad (5.14)$$

мұнда- ПО жалақының 10% - ын құрайтын және әлеуметтік салық салынбайтын зейнетақы жарналары:

$$ПО = З_{ФОТ} \cdot 10\%, \quad (5.15)$$

Материалдарға арналған шығындар сомасы бағдарламалық қамтамасыз етуді әзірлеу үшін қажетті тасымалдауыштарға, қағазға және басқа материалдарға көрсетілген шығындар негізге алына отырып есептеледі, мынадай формула бойынша айқындалады:

$$Р_м = \frac{З_0 * Н_м}{100\%}, \quad (5.16)$$

мұндағы $Н_м$ -базалық жалақыдан материалдарды жұмсау нормасы (3-5%). Арнайы жабдықтың ($Р_{со}$) құны шығындарды қоса алғанда, нақты бағдарламалық қамтылымды әзірлеу үшін қажетті көмекші арнайы жабдық пен бағдарламалық қамтылымды сатып алуға арналған қаражат шығындарын қамтиды:

$$P_{со} = \sum_{i=1}^{n} Ц_{оi}, \quad (5.17)$$

мұндағы $Ц_{оі}$ -нақты арнайы техниканың құны, мың теңге;

n -қолданылатын арнайы жабдықтың саны.

"Машина уақыты" ($Р_{мі}$) бабы бойынша шығыстар шешілетін міндеттердің сипатына және ДК типіне байланысты Машина уақытының бастапқы кодының ($Н_{мв}$) 100 жолына арналған стандарттарға сәйкес машина сағаттарында айқындалатын бағдарламалық қамтылымды әзірлеу және күйін келтіру үшін қажетті машина уақытын төлеуді қамтиды (Д қосымша):

$$Р_{мв} = Ц_{мв} * \left(\frac{V_0}{100}\right) * Н_{мв}, \quad (5.18)$$

мұндағы $Ц_{мв}$ -бір АВТО сағаттың бағасы, мың теңге;

$V_о$ -бағдарламалық қамтамасыз етудің жалпы көлемі (бастапқы код жолдары, LOC);

$Н_{мв}$ -бастапқы кодтың 100 жолын (машина сағаты) жөндеуге арналған машиналық уақыт шығынының стандарты.

Нақты бағдарламалық қамтамасыз ету үшін ғылыми іссапарларға (РНК) арналған шығыстар тұтастай ұйым үшін әзірленген стандартқа сәйкес немесе орындаушылардың базалық жалақысынан пайызбен айқындалады:

$$Р_{нк} = \frac{З_0 * Н_{рнк}}{100\%}, \quad (5.19)$$

мұнда $Н_{рнк}$ -жалпы ұйым бойынша іссапарларға арналған шығыстар нормативі.

Нақты бағдарламалық қамтылымға арналған "Өзге шығыстар" ($П_з$) бабы бойынша шығыстар мыналарды қамтиды

арнайы ғылыми-техникалық ақпаратты және арнайы әдебиетті сатып алуға арналған шығыстар. Жалпы ұйым үшін әзірленген стандартқа сәйкес базалық жалақының пайызымен анықталады:

$$П_з = \frac{З_0 * Н_{пз}}{100\%}, \quad (5.20)$$

мұндағы $Н_{Пз}$ -жалпы ұйым бойынша өзге де шығындар нормативі (%).

Үстеме шығындарға ($Р_{нак}$) басқарушы персоналды, қосалқы фермаларды және тәжірибелік қондырғыларды ұстау, сондай-ақ жалпы шаруашылық қажеттіліктерге арналған шығыстар кіреді:

$$Р_{нак} = \frac{З_0 * Н_{рн}}{100\%}, \quad (5.21)$$

мұндағы $H_{рн}$ -жалпы ұйым үшін стандартты үстеме шығыстар,%. 1-санаттағы ағымдағы айлық тарифтік мөлшерлеме ($T_{м1}$) тең болып қабылданады

50 мың теңге, тарифтік коэффициент 1,5, ФР = 8 сағатты құрайды.

Ай сайынғы тарифтік мөлшерлеме 5.11 формуласы бойынша есептеледі:

$$T_{\text{М}} = 50 \cdot 1{,}5 = 75 \text{ тыс. теңге.}$$

Сағаттық тарифтік ставканы есептеу 5.12 формула бойынша жүргізіледі:

$$\text{Т}_{\text{ч}} = \frac{75}{8 * 21} = 0{,}45 \text{ мың, тг.}$$

Қызметкердің жоспарланған жұмыс уақытының қоры ($Ф_{\text{п}}$) жобаны әзірлеу кезеңіне (38 күн) тең. Бонус коэффициенті (К) 10%, яғни 1,1 құрайды.

Әзірленген бағдарламалық қамтамасыз ету үшін қызметкердің базалық жалақысы 5.10 формула бойынша есептеледі:

$$З_{oi} = 0{,}45 \cdot 38 \cdot 8 \cdot 1{,}1 = 149{,}29 \text{ мың теңге.}$$

Қосымша жалақы базалық жалақының 10% - ын құрайды және 5.13.

$H_{\text{д}}$ = 10% формуласы бойынша есептеледі:

$$\text{З}_{\text{д}} = \frac{149{,}29 \cdot 10\%}{100\%} = 14{,}93 \text{ мын теңге.}$$

Құрылыс салушылардың жалақы қорының ($З_{ФОТ}$) мөлшері 5.9 формула бойынша есептеледі:

$$\text{ЗФОТ} = 149{,}29 + 14{,}93 = 164{,}21 \text{ мын теңге.}$$

Әлеуметтік салық қызметкердің табысынан 11%- ды құрайды (Қазақстан Республикасы Салық кодексінің 358-бабының 1-тармағы) және 5.14-формула бойынша есептеледі:

$$\text{З}_{\text{соц}} = (164{,}21 - 16{,}42) \cdot 11\% = 16{,}26 \text{ мын теңге.}$$

Мұнда *ПО* - жалақы мен әлеуметтік салықтың 10% - ын құрайтын зейнетақы жарналарына 5.15 формуласы бойынша есептелген салық салынбайды:

$$\text{ПО} = 164{,}21 \cdot 10\% = 16{,}42 \text{ мын теңге.}$$

Бастапқы деректер негізінде материалдарға жұмсалған шығындар сомасы 5.16. $H_{мз}$ = 3%формуласы бойынша айқындалады:

$$\text{Р}_{\text{м}} = \frac{149{,}29 * 3}{100\%} = 4.48 \text{ мын тг}$$

Арнайы жабдықтың ($P_{со}$) құны жұмыс станциясын сатып алу сомасын қамтиды, осы дипломдық жобада бір (n = 1) ДК пайдаланылған және 5.17 формуласы бойынша есептеледі:

$$P_{\text{со}} = 180 \text{ тыс. теңге.}$$

Машина уақытының құны ($P_{мв}$) 4.18, $H_{мв}$ = 15 (5.6-кесте), Vo = LOC = 1030 формуласы бойынша есептеледі, бір машина сағатының бағасы-оның электр энергиясына сұранысының құны, яғни.

$$\text{Ц}_{\text{мв}} = 500 * \frac{16{,}65}{10^3 * 10^3} = 0{,}01 \text{ мын тг.}$$

$$Р_{мв} = 0,1 * \frac{1030}{100} * 15 = 1,67 \text{ мын тг.}$$

Ғылыми сапарларға арналған шығыстар (Rnc) 5.19. nrnc = 30% формуласы бойынша есептеледі:

$$Р_{нк} = \frac{149,29 * 30\%}{100\%} = 44,79 \text{мын теңге.}$$

Өзге шығыстар бабы бойынша шығыстар 5.20. hpz = 20% формуласы бойынша есептеледі:

$$П_З = \frac{149,29*20\%}{100\%} = 29,86 \text{мын теңге}$$

Үстеме шығыстар (Rnak) 5.21. nrn = 70% формуласы бойынша есептеледі:

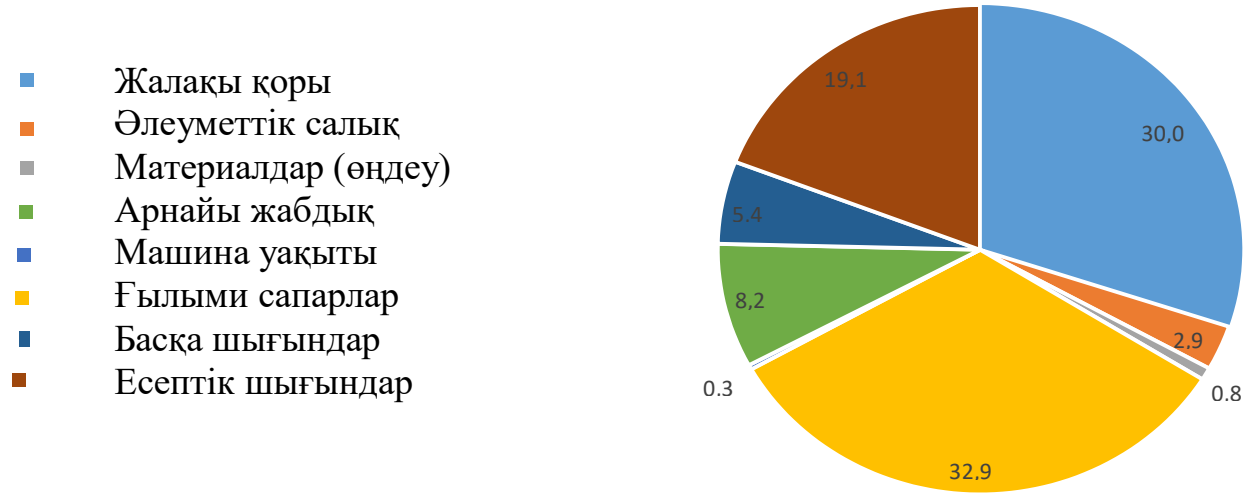
$$Р_{нак} = \frac{149,29*70\%}{100\%} = 104,5 \text{мын теңге}$$

Ақпараттық технологиялар түрінде жобалық шешімді әзірлеудің жалпы шығындарын есептеу 5.8 формула бойынша жүзеге асырылады:

$С_П$ = 164,21 + 16,26 + 4,48 + 180 + 1,67 + 44,79 + 29,86 + 104,5 =545,78 тыс теңге.

5.6-кесте-Әзірлеуге арналған шығындар

	Шартты белгі	Құны, теңге	Жалпы сомадан%
Жалақы қоры	*ЗФОТ*	164,21	30,09
Әлеуметтік салық	*Зсоц*	16,26	2,98
Материалдар (өңдеу)	*Рм*	4,48	0,82
Арнайы жабдық	*Рсо*	180	32,98
Машина уақыты	*Рмв*	1,68	0,31
Ғылыми сапарлар	*Рнк*	44,79	8,21
Басқа шығындар	*Пз*	29,86	5,47
Есептік шығындар	*Рнак*	104,5	19,15
Барлығы:		545,78 мың	100,00



5.1- сурет Әзірлеу шығындарының диаграммасы

Қорытынды

Дипломдық жобаны жазу нәтижесінде адамның толық бетін анықтау үшін мобильді қосымша жасалды.

Бағдарлама кең таралған, икемді және ыңғайлы бағдарламалау тіліне - Java 16 және Android SDK плагиніне негізделген. Сонымен қатар, Haar функциялары мен жергілікті екілік шаблондар беттің толық бетін анықтау үшін пайдаланылады, ал OpenCV компьютерлік көру кітапханасы осы әдістерді қолдану үшін пайдаланылады.

Мобильді қосымшаның интерфейсін мүмкіндігінше ыңғайлы және кәсіби түрде жасауға мүмкіндік беретін Android Development Tools технологиясы болды. Көптеген даму орталарының ішінде OpenCV - Eclipse IDE кітапханасымен ең тұрақты жұмыс таңдалды.

"Тіршілік қауіпсіздігі" бөлімінде жұмыс үй-жайындағы еңбек жағдайларына талдау, сондай-ақ ауаны баптау үшін қажетті параметрлерді есептеу жүргізілді.

Жасалған қосымша бағдарламалық жасақтаманы әзірлеудің барлық заманауи стандарттарына сәйкес келеді. Бағдарлама бетті тану әдісін қолданады, бұл құрылғыға жүктемені азайтады, бұл бағдарламаны кез-келген Android құрылғыларында пайдалануға мүмкіндік береді.

Дипломдық жоба барысында қойылған мақсатқа қол жеткізіліп, қойылған міндеттер орындалды.

Пайдаланылған әдебиеттер

1) Шилдт Г. Java 8. Полное руководство, 9-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 1376 с.

2) Прохоренок Н.А. OpenCV и Java. Обработка изображений и компью- терное зрение. – СПб.: БХВ – Петербург, 2018. – 320 с.

3) Дэвид А. Форсайт, Жан Понс. Компьютерное зрение. Современный подход, Издательство Вильямс, 2004. – 928 с.

4) Блох Д. Java. Эффективное программирование, Издательство Лори, 2014. – 310 с

5) Журавлева Л. В., Стригулин К. А. Исследования особенностей развития нейронных сетей в современном мире, Свое издательство, 2016. – 134 с.

6) Ставки налогов и обязательных платежей на 2018 год URL: https://online.zakon.kz/Document/?doc_id=37757559

7) Гонсалес Р., Вудс Р.Цифровая обработка изображений. Издание 3-е, исправленное и дополненное, Москва: Техносфера, 2012. – 1104 с.

8) Прэтт У. Цифровая обработка изображений: Пер.с англ. – М.: Мир, 1982. – Кн. 2 – 480 с.

9) Лукьяница А.А., Шишкин А.Г. Цифровая обработка видеоизображе- ний, М.: Ай-Эс-Эс Пресс, 2009. - 518 с.

10) Бёрд Б., Android Application Development All–in–One For Dummies. – Москва, 2011. - 816 с.

11) Дэрси Л. , Кондер Ш., Android за 24 часа. Программирование приложений под операционную систему Google – Москва, 2011. - 464 с.

12) Колисниченко Д., Программирование для Android. Самоучитель. – Москва, 2012. - 272 с.

13) Майер Р., Android 4. Программирование приложений для планшетных компьютеров и смартфонов – Москва, 2013. - 816 с.

14) Медникс З., Дорнин Л., Мик Б., Накамура М., Программирование под Android – Москва, 2013. - 560 с.

15) Android // wikipedia.org URL: https://ru.wikipedia.org/wiki/Android (дата обращения: 01.05.2017).

16) Android Runtime// wikipedia.org URL: https://ru.wikipedia.org/wiki/Android_Runtime (дата обращения: 01.05.2017).

17) Download Android Studio and SDK Tools // developer.android.com URL: https://developer.android.com/studio/index.html (дата

18) Dashboard | Android Developers // developer.android.com URL: https://developer.android.com/about/dashboards/index.html (дата обращения: 05.05.2017).

А-қосымшасы. Тұлғаны тануға арналған мобильді қосымша

```
package ch.zhaw.facerecognition.Activities;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import java.io.File;
import ch.zhaw.facerecognitionlibrary.Helpers.FileHelper;
import ch.zhaw.facerecognition.R;

public class MainActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

Intent intent = getIntent();
String training = intent.getStringExtra("training");
if (training != null && !training.isEmpty()){
Toast.makeText(getApplicationContext(), training,
Toast.LENGTH_SHORT).show();
intent.removeExtra("training");
}

double accuracy = intent.getDoubleExtra("accuracy", 0);
if (accuracy != 0){
Toast.makeText(getApplicationContext(), "The accuracy was " + accuracy *
100 + " %", Toast.LENGTH_LONG).show();
intent.removeExtra("accuracy");
}

PreferenceManager.setDefaultValues(this, R.xml.preferences, false);

Button callSettings = (Button)findViewById(R.id.button_settings);
callSettings.setOnClickListener(new View.OnClickListener() {
@Override
```

```
        public void onClick(View v) {
        startActivity(new Intent(v.getContext(), SettingsActivity.class));
        }
        });

        Button callAddPerson = (Button)findViewById(R.id.button_addPerson);
        callAddPerson.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        startActivity(new Intent(v.getContext(), AddPersonActivity.class));
        }
        });

        FileHelper fh = new FileHelper();

        Button callDetectionTest =
(Button)findViewById(R.id.button_detection_test);
        if(fh.getDetectionTestList().length == 0) callDetectionTest.setEnabled(false);
        callDetectionTest.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        startActivity(new Intent(v.getContext(), DetectionTestActivity.class));
        }
        });

        Button callDetectionView =
(Button)findViewById(R.id.button_detection_view);
        callDetectionView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        startActivity(new Intent(v.getContext(), DetectionActivity.class));
        }
        });




        package ch.zhaw.facerecognition.Activities;
        import android.app.Activity;
        import android.content.Intent;
        import android.os.Bundle;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ToggleButton;

import java.io.File;

import ch.zhaw.facerecognitionlibrary.Helpers.FileHelper;
import ch.zhaw.facerecognition.R;

public class AddPersonActivity extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_add_person);

final ToggleButton btnTrainingTest =
(ToggleButton)findViewById(R.id.btnTrainingTest);
final ToggleButton btnReferenceDeviation =
(ToggleButton)findViewById(R.id.btnReferenceDeviation);
final ToggleButton btnTimeManually =
(ToggleButton)findViewById(R.id.btnTimeManually);
btnTrainingTest.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
if(btnTrainingTest.isChecked()){
btnReferenceDeviation.setEnabled(true);
} else {
btnReferenceDeviation.setEnabled(false);
}
}
});

Button btn_Start = (Button)findViewById(R.id.btn_Start);
btn_Start.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
EditText txt_Name = (EditText)findViewById(R.id.txt_Name);
String name = txt_Name.getText().toString();
Intent intent = new Intent(v.getContext(), AddPersonPreviewActivity.class);
```

```
intent.putExtra("Name", name);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

if(btnTimeManually.isChecked()){
intent.putExtra("Method", AddPersonPreviewActivity.MANUALLY);
} else {
intent.putExtra("Method", AddPersonPreviewActivity.TIME);
}

if(btnTrainingTest.isChecked()){
// Add photos to "Test" folder
if(isNameAlreadyUsed(new FileHelper().getTestList(), name)){
Toast.makeText(getApplicationContext(), "This name is already used. Please
choose another one.", Toast.LENGTH_SHORT).show();
} else {
intent.putExtra("Folder", "Test");
if(btnReferenceDeviation.isChecked()){
intent.putExtra("Subfolder", "deviation");
} else {
intent.putExtra("Subfolder", "reference");
}
startActivity(intent);
}
} else {
// Add photos to "Training" folder

if(isNameAlreadyUsed(new FileHelper().getTrainingList(), name)){
Toast.makeText(getApplicationContext(), "This name is already used. Please
choose another one.", Toast.LENGTH_SHORT).show();
} else {
intent.putExtra("Folder", "Training");
startActivity(intent);
}
}
}
});
}
private boolean isNameAlreadyUsed(File[] list, String name){
boolean used = false;
if(list != null && list.length > 0){
for(File person : list){
// The last token is the name --> Folder name = Person name
```

```
String[] tokens = person.getAbsolutePath().split("/");
final String foldername = tokens[tokens.length-1];
if(foldername.equals(name)){
used = true;
break;
}
}
}
return used;
}
}
//*************************************************************

package ch.zhaw.facerecognition.Activities;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.os.Bundle;
import android.view.SurfaceView;
import android.view.View;
import android.widget.ImageButton;

import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Rect;

import java.io.File;
import java.util.Date;
import java.util.List;

import ch.zhaw.facerecognitionlibrary.Helpers.CustomCameraView;
import ch.zhaw.facerecognitionlibrary.Helpers.FileHelper;
import ch.zhaw.facerecognitionlibrary.Helpers.MatName;
import ch.zhaw.facerecognitionlibrary.Helpers.MatOperation;
import ch.zhaw.facerecognitionlibrary.PreProcessor.PreProcessorFactory;
import ch.zhaw.facerecognition.R;
```

```
public class AddPersonPreviewActivity extends Activity implements
CameraBridgeViewBase.CvCameraViewListener2 {
public static final int TIME = 0;
public static final int MANUALLY = 1;
private CustomCameraView mAddPersonView;
// The timerDiff defines after how many milliseconds a picture is taken
private long timerDiff;
private long lastTime;
private PreProcessorFactory ppF;
private FileHelper fh;
private String folder;
private String subfolder;
private String name;
private int total;
private int numberOfPictures;
private int method;
private ImageButton btn_Capture;
private boolean capturePressed;
private boolean front_camera;
private boolean night_portrait;
private int exposure_compensation;

static {
if (!OpenCVLoader.initDebug()) {
// Handle initialization error
}
}

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_add_person_preview);

Intent intent = getIntent();
folder = intent.getStringExtra("Folder");
if(folder.equals("Test")){

subfolder = intent.getStringExtra("Subfolder");
}
name = intent.getStringExtra("Name");
method = intent.getIntExtra("Method", 0);
capturePressed = false;
```

```
        if(method == MANUALLY){
        btn_Capture = (ImageButton)findViewById(R.id.btn_Capture);
        btn_Capture.setVisibility(View.VISIBLE);
        btn_Capture.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        capturePressed = true;
        }
        });
        }

        fh = new FileHelper();
        total = 0;
        lastTime = new Date().getTime();

        SharedPreferences sharedPrefs =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        timerDiff = Integer.valueOf(sharedPrefs.getString("key_timerDiff", "500"));

        mAddPersonView = (CustomCameraView)
findViewById(R.id.AddPersonPreview);
        // Use camera which is selected in settings
        SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(this);
        front_camera = sharedPref.getBoolean("key_front_camera", true);

        numberOfPictures =
Integer.valueOf(sharedPref.getString("key_numberOfPictures", "100"));

        night_portrait = sharedPref.getBoolean("key_night_portrait", false);
        exposure_compensation =
Integer.valueOf(sharedPref.getString("key_exposure_compensation", "50"));

        if (front_camera){
        mAddPersonView.setCameraIndex(CameraBridgeViewBase.CAMERA_ID_
FRONT);
        } else {
        mAddPersonView.setCameraIndex(CameraBridgeViewBase.CAMERA_ID_
BACK);
        }
        mAddPersonView.setVisibility(SurfaceView.VISIBLE);
        mAddPersonView.setCvCameraViewListener(this);
```

```
        int maxCameraViewWidth =
Integer.parseInt(sharedPref.getString("key_maximum_camera_view_width",
"640"));
        int maxCameraViewHeight =
Integer.parseInt(sharedPref.getString("key_maximum_camera_view_height",
"480"));
        mAddPersonView.setMaxFrameSize(maxCameraViewWidth,
maxCameraViewHeight);
        }

        @Override
        public void onCameraViewStarted(int width, int height) {

        if (night_portrait) {
        mAddPersonView.setNightPortrait();
        }

        if (exposure_compensation != 50 && 0 <= exposure_compensation &&
exposure_compensation <= 100)
        mAddPersonView.setExposure(exposure_compensation);
        }

        @Override
        public void onCameraViewStopped() {

        }

        @Override
        public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
        Mat imgRgba = inputFrame.rgba();
        Mat imgCopy = new Mat();
        imgRgba.copyTo(imgCopy);
        // Selfie / Mirror mode
        if(front_camera){
        Core.flip(imgRgba,imgRgba,1);
        }
```

```
        long time = new Date().getTime();
        if((method == MANUALLY) || (method == TIME) && (lastTime +
timerDiff < time)){
        lastTime = time;

        // Check that only 1 face is found. Skip if any or more than 1 are found.
        List<Mat> images = ppF.getCroppedImage(imgCopy);
        if (images != null && images.size() == 1){
        Mat img = images.get(0);
        if(img != null){
        Rect[] faces = ppF.getFacesForRecognition();
        //Only proceed if 1 face has been detected, ignore if 0 or more than 1 face
have been detected
        if((faces != null) && (faces.length == 1)){
        faces = MatOperation.rotateFaces(imgRgba, faces,
ppF.getAngleForRecognition());
        if(((method == MANUALLY) && capturePressed) || (method == TIME)){
        MatName m = new MatName(name + "_" + total, img);
        if (folder.equals("Test")) {
        String wholeFolderPath = fh.TEST_PATH + name + "/" + subfolder;
        new File(wholeFolderPath).mkdirs();
        fh.saveMatToImage(m, wholeFolderPath + "/");
        } else {
        String wholeFolderPath = fh.TRAINING_PATH + name;
        new File(wholeFolderPath).mkdirs();
        fh.saveMatToImage(m, wholeFolderPath + "/");
        }

        for(int i = 0; i<faces.length; i++){
        MatOperation.drawRectangleAndLabelOnPreview(imgRgba, faces[i],
String.valueOf(total), front_camera);
        }

        total++;

        // Stop after numberOfPictures (settings option)
        if(total >= numberOfPictures){
        Intent intent = new Intent(getApplicationContext(),
AddPersonActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
```

```
startActivity(intent);
}
capturePressed = false;
} else {
for(int i = 0; i<faces.length; i++){
MatOperation.drawRectangleOnPreview(imgRgba, faces[i], front_camera);
}
}
}
}
}
}

return imgRgba;
}

@Override
public void onResume()
{
super.onResume();

ppF = new PreProcessorFactory(this);
mAddPersonView.enableView();
}

@Override
public void onPause()
{
super.onPause();
if (mAddPersonView != null)
mAddPersonView.disableView();
}

public void onDestroy() {
super.onDestroy();
if (mAddPersonView != null)
mAddPersonView.disableView();
}
}

//*****************************************************************
```

```
package ch.zhaw.facerecognition.Activities;

import android.content.res.Configuration;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.support.annotation.LayoutRes;
import android.support.annotation.Nullable;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatDelegate;
import android.support.v7.widget.Toolbar;
import android.view.MenuInflater;
import android.view.View;
import android.view.ViewGroup;

/**
* A {@link android.preference.PreferenceActivity} which implements and
proxies the necessary calls
* to be used with AppCompat.
*/
public abstract class AppCompatPreferenceActivity extends
PreferenceActivity {

private AppCompatDelegate mDelegate;

@Override
protected void onCreate(Bundle savedInstanceState) {
getDelegate().installViewFactory();
getDelegate().onCreate(savedInstanceState);
super.onCreate(savedInstanceState);
}

@Override
protected void onPostCreate(Bundle savedInstanceState) {
super.onPostCreate(savedInstanceState);
getDelegate().onPostCreate(savedInstanceState);
}

public ActionBar getSupportActionBar() {
return getDelegate().getSupportActionBar();
}

public void setSupportActionBar(@Nullable Toolbar toolbar) {
```

```
getDelegate().setSupportActionBar(toolbar);
}

@Override
public MenuInflater getMenuInflater() {
return getDelegate().getMenuInflater();
}

@Override
public void setContentView(@LayoutRes int layoutResID) {
getDelegate().setContentView(layoutResID);
}

@Override
public void setContentView(View view) {
getDelegate().setContentView(view);
}

@Override
public void setContentView(View view, ViewGroup.LayoutParams params)
{
getDelegate().setContentView(view, params);
}

@Override
public void addContentView(View view, ViewGroup.LayoutParams params)
{
getDelegate().addContentView(view, params);
}

@Override
protected void onPostResume() {
super.onPostResume();
getDelegate().onPostResume();
}

@Override
protected void onTitleChanged(CharSequence title, int color) {
super.onTitleChanged(title, color);
getDelegate().setTitle(title);
}
```

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
super.onConfigurationChanged(newConfig);
getDelegate().onConfigurationChanged(newConfig);
}

@Override
protected void onStop() {
super.onStop();
getDelegate().onStop();
}

@Override
protected void onDestroy() {
super.onDestroy();
getDelegate().onDestroy();
}

public void invalidateOptionsMenu() {
getDelegate().invalidateOptionsMenu();
}

private AppCompatDelegate getDelegate() {
if (mDelegate == null) {
mDelegate = AppCompatDelegate.create(this, null);
}
return mDelegate;
}
}
//**************************************************************
package ch.zhaw.facerecognition.Activities;

import android.app.Activity;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.SurfaceView;
import android.view.WindowManager;

import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
```

```
import org.opencv.core.Mat;
import org.opencv.core.Rect;
import java.util.List;

import ch.zhaw.facerecognition.R;
import ch.zhaw.facerecognitionlibrary.Helpers.CustomCameraView;
import ch.zhaw.facerecognitionlibrary.Helpers.MatOperation;
import ch.zhaw.facerecognitionlibrary.PreProcessor.PreProcessorFactory;

public class DetectionActivity extends Activity implements
CameraBridgeViewBase.CvCameraViewListener2 {
private boolean front_camera;
private boolean night_portrait;
private int exposure_compensation;
private CustomCameraView mDetectionView;
private PreProcessorFactory ppF;

static {
if (!OpenCVLoader.initDebug()) {
// Handle initialization error
}
}

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCR
EEN_ON);
setContentView(R.layout.activity_detection);

mDetectionView = (CustomCameraView)
findViewById(R.id.DetectionView);
// Use camera which is selected in settings
SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(this);
front_camera = sharedPref.getBoolean("key_front_camera", true);
night_portrait = sharedPref.getBoolean("key_night_portrait", false);
exposure_compensation =
Integer.valueOf(sharedPref.getString("key_exposure_compensation", "20"));

if (front_camera){
```

```
        mDetectionView.setCameraIndex(CameraBridgeViewBase.CAMERA_ID_F
RONT);
        } else {
        mDetectionView.setCameraIndex(CameraBridgeViewBase.CAMERA_ID_B
ACK);
        }
        mDetectionView.setVisibility(SurfaceView.VISIBLE);
        mDetectionView.setCvCameraViewListener(this);

        int maxCameraViewWidth =
Integer.parseInt(sharedPref.getString("key_maximum_camera_view_width",
"640"));
        int maxCameraViewHeight =
Integer.parseInt(sharedPref.getString("key_maximum_camera_view_height",
"480"));
        mDetectionView.setMaxFrameSize(maxCameraViewWidth,
maxCameraViewHeight);
        }

        @Override
        public void onCameraViewStarted(int width, int height) {
        if (night_portrait) {
        mDetectionView.setNightPortrait();
        }

        if (exposure_compensation != 50 && 0 <= exposure_compensation &&
exposure_compensation <= 100)
        mDetectionView.setExposure(exposure_compensation);
        }

        @Override
        public void onCameraViewStopped() {

        }
        @Override
        public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
        Mat imgRgba = inputFrame.rgba();
        Mat img = new Mat();
        imgRgba.copyTo(img);
        List<Mat> images = ppF.getCroppedImage(img);
        Rect[] faces = ppF.getFacesForRecognition();
```

```
        // Selfie / Mirror mode
        if(front_camera){
        Core.flip(imgRgba,imgRgba,1);
        }
        if(images == null || images.size() == 0 || faces == null || faces.length == 0 || !
(images.size() == faces.length)){
        // skip
        return imgRgba;
        } else {
        faces = MatOperation.rotateFaces(imgRgba, faces,
ppF.getAngleForRecognition());
        for(int i = 0; i<faces.length; i++){
        MatOperation.drawRectangleAndLabelOnPreview(imgRgba, faces[i], "",
front_camera);
        }
        return imgRgba;
        }
        }

        @Override
        protected void onResume() {
        super.onResume();
        ppF = new PreProcessorFactory(getApplicationContext());
        mDetectionView.enableView();
        }
        }
        //**************************************************************
        package ch.zhaw.facerecognition.Activities;

        import android.content.Intent;
        import android.os.Handler;
        import android.os.Looper;
        import android.preference.PreferenceManager;
        import android.support.v7.app.AppCompatActivity;
        import android.os.Bundle;
        import android.text.method.ScrollingMovementMethod;
        import android.widget.TextView;

        import org.opencv.android.OpenCVLoader;
        import org.opencv.core.Mat;
        import org.opencv.core.Rect;
        import org.opencv.imgcodecs.Imgcodecs;
```

```
import org.opencv.imgproc.Imgproc;
import java.io.File;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Map;

import ch.zhaw.facerecognition.R;
import ch.zhaw.facerecognitionlibrary.Helpers.FileHelper;
import ch.zhaw.facerecognitionlibrary.Helpers.MatName;
import ch.zhaw.facerecognitionlibrary.Helpers.MatOperation;
import ch.zhaw.facerecognitionlibrary.Helpers.PreferencesHelper;
import ch.zhaw.facerecognitionlibrary.PreProcessor.PreProcessorFactory;

public class DetectionTestActivity extends AppCompatActivity {
private static final String RESULT_NEGATIVE = "negative";
private static final String RESULT_POSITIVE = "positive";
TextView progress;
Thread thread;

static {
if (!OpenCVLoader.initDebug()) {
// Handle initialization error
}
}

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_detection_test);
progress = (TextView) findViewById(R.id.progressText);
progress.setMovementMethod(new ScrollingMovementMethod());
}
@Override
protected void onResume() {
super.onResume();
final Handler handler = new Handler(Looper.getMainLooper());
thread = new Thread(new Runnable() {
public void run() {
if(!Thread.currentThread().isInterrupted()){
PreProcessorFactory ppF = new
PreProcessorFactory(getApplicationContext());
```

```
        FileHelper fileHelper = new FileHelper();
        File[] detectionFolders = fileHelper.getDetectionTestList();
        if (detectionFolders.length > 0) {
        // total and matches are used to calculate the accuracy afterwards
        int total = 0;
        int matches = 0;
        List<String> results = new ArrayList<>();
        results.add("Expected Name;Expected File;Result");
        Date time_start = new Date();
        for (File folder : detectionFolders) {
        File[] files = folder.listFiles();
        int counter = 1;
        for (File file : files) {
        if (FileHelper.isFileAnImage(file)) {
        Mat imgRgba = Imgcodecs.imread(file.getAbsolutePath());
        Imgproc.cvtColor(imgRgba, imgRgba, Imgproc.COLOR_BGRA2RGBA);

        List<Mat> images = ppF.getProcessedImage(imgRgba,
PreProcessorFactory.PreprocessingMode.DETECTION);
        Rect[] faces = ppF.getFacesForRecognition();

        String result = "";

        if (faces == null || faces.length == 0) {
        result = RESULT_NEGATIVE;
        } else {
        result = RESULT_POSITIVE;
        faces = MatOperation.rotateFaces(imgRgba, faces,
ppF.getAngleForRecognition());
        for(int i = 0; i<faces.length; i++){
        MatOperation.drawRectangleAndLabelOnPreview(images.get(0), faces[i], "",
false);
        }
        }

        // Save images
        String[] tokens = file.getName().split("\\.");
        String filename = tokens[0];
        for (int i=0; i<images.size();i++){
        MatName m = new MatName(filename + "_" + (i + 1), images.get(i));
```

```
        fileHelper.saveMatToImage(m, FileHelper.RESULTS_PATH + "/" +
time_start.toString() + "/");
        }

        tokens = file.getParent().split("/");
        final String name = tokens[tokens.length - 1];

        results.add(name + ";" + file.getName() + ";" + result);

        total++;

        if (name.equals(result)) {
        matches++;
        }
        // Update screen to show the progress
        final int counterPost = counter;
        final int filesLength = files.length;
        progress.post(new Runnable() {
        @Override
        public void run() {
        progress.append("Image " + counterPost + " of " + filesLength + " from " +
name + "\n");
        }
        });
        counter++;
        }
        }
        }
        Date time_end = new Date();
        long duration = time_end.getTime() - time_start.getTime();
        int durationPerImage = (int) duration / total;
        double accuracy = (double) matches / (double) total;
        Map<String, ?> printMap =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext()).getAll()
;
        fileHelper.saveResultsToFile(printMap, accuracy, durationPerImage, results);

        final Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.putExtra("accuracy", accuracy);
        handler.post(new Runnable() {
```

```
@Override
public void run() {
startActivity(intent);
}
});
}
} else {
Thread.currentThread().interrupt();
}
}
});
thread.start();
}

@Override
protected void onPause() {
super.onPause();
thread.interrupt();
}

@Override
protected void onStop() {
super.onStop();
thread.interrupt();
}
}
//*************************************************************

package ch.zhaw.facerecognition.Activities;
import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Looper;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.SurfaceView;
import android.view.View;
import android.view.WindowManager;
import android.widget.ProgressBar;

import org.opencv.android.CameraBridgeViewBase;
```

```
import org.opencv.android.OpenCVLoader;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Rect;
import java.io.File;
import java.util.List;
import ch.zhaw.facerecognitionlibrary.Helpers.CustomCameraView;
import ch.zhaw.facerecognitionlibrary.Helpers.FileHelper;
import ch.zhaw.facerecognitionlibrary.Helpers.MatOperation;
import ch.zhaw.facerecognitionlibrary.Helpers.PreferencesHelper;
import ch.zhaw.facerecognitionlibrary.PreProcessor.PreProcessorFactory;
import ch.zhaw.facerecognition.R;
import ch.zhaw.facerecognitionlibrary.Recognition.Recognition;
import ch.zhaw.facerecognitionlibrary.Recognition.RecognitionFactory;

public class RecognitionActivity extends Activity implements
CameraBridgeViewBase.CvCameraViewListener2 {
private CustomCameraView mRecognitionView;
private static final String TAG = "Recognition";
private FileHelper fh;
private Recognition rec;
private PreProcessorFactory ppF;
private ProgressBar progressBar;
private boolean front_camera;
private boolean night_portrait;
private int exposure_compensation;

static {
if (!OpenCVLoader.initDebug()) {
// Handle initialization error
}
}

@Override
public void onCreate(Bundle savedInstanceState) {
Log.i(TAG,"called onCreate");
super.onCreate(savedInstanceState);
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCR
EEN_ON);
```

```
        setContentView(R.layout.recognition_layout);
        progressBar = (ProgressBar)findViewById(R.id.progressBar);
        fh = new FileHelper();
        File folder = new File(fh.getFolderPath());
        if(folder.mkdir() || folder.isDirectory()){
        Log.i(TAG,"New directory for photos created");
        } else {
        Log.i(TAG,"Photos directory already existing");
        }
        mRecognitionView = (CustomCameraView)
findViewById(R.id.RecognitionView);
        // Use camera which is selected in settings
        SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(this);
        front_camera = sharedPref.getBoolean("key_front_camera", true);
        night_portrait = sharedPref.getBoolean("key_night_portrait", false);
        exposure_compensation =
Integer.valueOf(sharedPref.getString("key_exposure_compensation", "20"));

        if (front_camera){
        mRecognitionView.setCameraIndex(CameraBridgeViewBase.CAMERA_ID
_FRONT);
        } else {
        mRecognitionView.setCameraIndex(CameraBridgeViewBase.CAMERA_ID
_BACK);
        }
        mRecognitionView.setVisibility(SurfaceView.VISIBLE);
        mRecognitionView.setCvCameraViewListener(this);

        int maxCameraViewWidth =
Integer.parseInt(sharedPref.getString("key_maximum_camera_view_width",
"640"));
        int maxCameraViewHeight =
Integer.parseInt(sharedPref.getString("key_maximum_camera_view_height",
"480"));
        mRecognitionView.setMaxFrameSize(maxCameraViewWidth,
maxCameraViewHeight);
        }

        @Override
        public void onPause()
        {
```

```
        super.onPause();
        if (mRecognitionView != null)

        mRecognitionView.disableView();
        }

        public void onDestroy() {
        super.onDestroy();
        if (mRecognitionView != null)
        mRecognitionView.disableView();
        }

        public void onCameraViewStarted(int width, int height) {

        if (night_portrait) {
        mRecognitionView.setNightPortrait();
        }

        if (exposure_compensation != 50 && 0 <= exposure_compensation &&
exposure_compensation <= 100)
        mRecognitionView.setExposure(exposure_compensation);
        }

        public void onCameraViewStopped() {
        }

        public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
        Mat imgRgba = inputFrame.rgba();
        Mat img = new Mat();
        imgRgba.copyTo(img);
        List<Mat> images = ppF.getProcessedImage(img,
PreProcessorFactory.PreprocessingMode.RECOGNITION);
        Rect[] faces = ppF.getFacesForRecognition();

        // Selfie / Mirror mode
        if(front_camera){
        Core.flip(imgRgba,imgRgba,1);
        }
        if(images == null || images.size() == 0 || faces == null || faces.length == 0 || !
(images.size() == faces.length)){
        // skip
```

```
        return imgRgba;
        } else {

        faces = MatOperation.rotateFaces(imgRgba, faces,
ppF.getAngleForRecognition());
        for(int i = 0; i<faces.length; i++){
        MatOperation.drawRectangleAndLabelOnPreview(imgRgba, faces[i],
rec.recognize(images.get(i), ""), front_camera);
        }
        return imgRgba;
        }
        }

        @Override
        public void onResume()
        {
        super.onResume();

        ppF = new PreProcessorFactory(getApplicationContext());

        final android.os.Handler handler = new
android.os.Handler(Looper.getMainLooper());
        Thread t = new Thread(new Runnable() {
        public void run() {
        handler.post(new Runnable() {
        @Override
        public void run() {
        progressBar.setVisibility(View.VISIBLE);

        }
        });
        SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        String algorithm = sharedPref.getString("key_classification_method",
getResources().getString(R.string.eigenfaces));
        rec = RecognitionFactory.getRecognitionAlgorithm(getApplicationContext(),
Recognition.RECOGNITION, algorithm);
        handler.post(new Runnable() {
        @Override
        public void run() {
        progressBar.setVisibility(View.GONE);
        }
```

```
});
}
});


t.start();

// Wait until Eigenfaces loading thread has finished
try {
t.join();
} catch (InterruptedException e) {
e.printStackTrace();
}

mRecognitionView.enableView();
}
}
```