**Բովանդակություն**

**Ներածություն** ………………………………………………………………………...……………4

**Գլուխ 1. Գրականության նյութերի ամփոփում** ……………………………..………………6

1.1 Պիկսել և գույն…………………………………..…………………………………..6

1.2 Ռաստերային պատկերները և նրանց հիմնական

բնութագրությունները………………………………………………….………..…8

1.3 Գույների քանակը ………………………………………………………………...10

1.4 BMP ֆորմատի կառուցվածքը ………………………………………..................11

1.5 JPEG գրաֆիկական ձևաչափ…………………………………………….............14

1.6 AVI ֆորմատի կառուցվածքը……………………………………………............23

**Գլուխ 2. Խնդրի դրվածքը և կատարված աշխատանքի ալգորիթմի նկարագրությունը** ……………………………………………………………………………………………………….24

2.1. Խնդրի դրվածք….…………………………………………………………………..24

2.2. Ալգորիթմի նկարագրությունը ……………………………………………….….30

**Գլուխ 3. Ծրագրի աշխատանքը**..……………………………………………………………....35

**Գլուխ 4. Ծրագրի տեխնիկատնտեսական հիմնավորում** ..…………………….…………....48

**Գլուխ 5. Կենսագործունեության անվտանգություն** ……………………………...……….....59

**Գլուխ 6. Բնապահպանության հարցերի հիմնավորում** …………………………………...66

**Եզրակացություն** ………………………………………………………………………..……….75

**Գրականություն** ………………………………………………………………………………....76

**Հավելված** ……………………………………………………………………………………...…77

**Ներածություն**

Մարդը իր գոյության ամբողջ ընթացքում ձգտել է հեշտացնել իր առօրյա կյանքը: Այդ ուղղությամբ հիմնական քայլերից է հանդիսացել համակարգչի ստեղծումը: Համակարգիչները լուծում են հիմիկվա խնդիրների ահռելի մեծ բազմություն և որպես այդ բազմության մի տարր կարելի է նշել գրաֆիկական պատկերների մշակումը, ստեղծումն ու փոփոխումը: Պատկերների հետ կապված համակարգչային ուղղությունները բազմաթիվ են, փորձենք նշել դրանցից մի քանիսը՝

* Համակարգչային գրաֆիկա
* Պատկերների և վիդեո հոսքերի մշակում

Համակարգչային գրաֆիկայի համար կան բազմաթիվ լուծումներ որոնք արդեն իրենց կայուն տեղնեն գտել ժամանակակից շուկայում: Ամենահայտնի լուծումներից կարելի է նշել Adobe Creative Suite պատկերների և վիդեոների մշակման պրոֆեսիոնալ ծրագրերի փաթեթը և Corel Draw գծագրեր և պատկերներ ստեղծելու և մշակելու համար նախատեսված ծրագիրը: Սակայն շուկան շատ սուղ է վերը նշված 2-րդ կետում: Այդ պատճառով տվյալ աշխատանքում հիմնականում կանրադառնանք 2-րդ կետում նշված խնդիրների լուզմանը, և ծրագրային ապահովման մշակմանը:

Ծրագրային ապահովումը գրվել է օգտագործելով Visual Studio 2010 Ultimate ծրագրային փաթեթը: Ծրագիրը ամբողջովին գրված է C# ծրագրավորման լեզվով: Ծրագրի իրագործման համար օգտագործվել է OpenCV գրադարանի C#-ի համար համապատասխանեցված EmguCV տարբերակը:

C# ծրագրավորման լեզուն հանդիսանում է լիովին օբյեկտա-կողմնորոշված լեզու և ունենալով շատ մեծ հնարավորություններ շատ պարզ և ինտուիտիվ հասկանալի է ցանկացած մակարդակի ծրագրավորողների համար: C#-ով գրված բոլոր ծրագրերը սկզբից կոմպիլացվում են բայտ կոդի որն այնուհետև կատարվում է CLR-ի կողմից (Common Language Runtime): C#-ը հանդիսանում է արագ կատարվող կոդ կիրառական ծրագրերի համար և հիմնական պահանջները ապահովելու համար անհրաժեշտ ժամանակահատվածը համեմատելի է C++-ով կամ C-ով գրված ծրագրերի հետ: Սակայն C#-ով անհնար է գրել կատարյալ օպերացիոն համարգ, որը կհամապատասխանի հիմիկվա օպերացիոն համակարգերի առաջ դրված բոլոր պահանջներին:

Այս խնդրի լուծման համար գրված ծրագրի շրջանակներում C# ծրագրավորման լեզուն լիովին բավարար է:

**ԳԼՈՒԽ 1**

**Գրականության նյութերի ամփոփում**

* 1. **Պիկսել և գույն**

Հիմնական “աղյուսակը”, որից կառուցվում են բոլոր համակարգչային պատկերները, համարվում է նկարի տարրը կամ պիկսելը: Պիկսելը դա լույսի բիծ է դիսպլեյի էկրանի վրա, որը կարող է ընդունել տարբեր գուներանգներ: Յուրաքանչյուր պատկեր անկախ իր բարդությունից, դա ընդամենը պիկսելների ամբողջություն է: Համակարգչի համար գլխավոր խորամանկությունը կայանում է նրանում որ համապատասխան ներկերը դրվեն անհրաժեշտ տեղերում:

Պիկսելները էկրանի վրա զետեղվում են տողերի և սյունակերի ձևով: Էկրանի թույլտվությունը որոշվում է պիկսելների քանակով յուրաքանչյուր տողում կամ յուրաքանչյուր սյունակում:

Օրինակ 640 x 480 թույլատվությամբ էկրանը հնարավորություն է տալիս պատկերել 640 պիկսել հորիզոնաբար և 480 պիկսել ուղղահայաց: Առաջին հայացքից պիկսելների քանակը այդքան շատ չէ: Բայց բազմապատկենք 640–ը 480-ով և կպարզենք որ էկրանի վրա տեղավորվում է 307.200 պիկսել: Իսկ դա շատ է: Այժմյան պատկերացումներով 640 x 480 թույլտվությունը համարվում է ցածր: Արդեն օգտագորշվում են ավելի բարձր թույլտվություններ օրինակ 1024 x 768 և ավելի բարձր:

Յուրաքանչյուր պիկսելի գույնի փոփոխությունը կարելի է անել անկախ, սակայն երանգների քանակը, որոնք կարող են միաժամանկ գտնվել էկրանի վրա, սահմանափակված են և կախված են օգտագործվող գրաֆիկական սարքավորումից: Էկրանի վրա միաժամանակ արտացոլվող գույնորի մաքսիմալ քանակը որոշվում է վիդեոբուֆերում յուրաքանչյուր պիկսելի համար հատկացված բիտերի քանակով: Լիագույն համակարգերում յուրաքանչյուր գույնին հատկացվում է 24 բիտ տեղեկությունª 8 կարմիրի համար, 8 կանաչի, 8 կապույտի: 8 բիտ ինֆորմացիան կարող է պարունակել 0-255 թվերը: 24 բիտ ունեցող թիվը կարող է գտնվել 0 – 16.777.215 սահմններում: Դա նշանակում է որ վիդեոադապտերը կարող է պատկերել 16.7 մլն. գույնից ավելին:Խառնելով կարմիր, կանաչ և կապույտ գույների տարբեր ինտենսիվություններ կարելի է ստանալ ցանկացած գույն:

Վիդեոբուֆերի բիտերը, որոնք համապատասխանում են այս կամ այն պիկսելին, չեն մատնանշում անմիջապես նրա գույնը: 256 գույն ունեցող համակարգում արժեքը վիդեոբուֆերից մատնաշում է աղյուսակի 256 տողերից մեկի վրա, որը կոչվում է գունային ներկապնակ: Ներկապնակի այդ տողի վրա գտնվող թիվը որոշում է պիկսելի գույնը:

Որպես օրենք, որքան շատ են էկրանի վրա պիկսելները, այնքան բարձր է պատկերի որակը: Հաճախ օգտագործողը գտնվում է ընտրության առջևª ավելի շատ գույներ թե ավելի մեծ թույլտվություն: Միևնույն ադապտերը թույլ է տալիս ստանալ 256 գույն 1024x768 թույլտվության դեպքում, բայց 16 գույն 1280x1024 թույլատվության դեպքում: Այսպիսով ինչն է կարևոր ª ավելի մեծ թույլտվությունը թե ավելի շատ երանգները: Եթե էկրանին պետք է ստանալ լուսանկարչական որակի պատկեր, ապա ավելի կարևվոր են երանգները: Փոքր թույլատվություն ունեցող պատկերը, որը պարունակում է 256 գույն, ավելի իրական տեսք ունի , քան ավելի մեծ թույլատվություն ունեցող պատկերը: Այդ պատճառով էլ հեռուստացույցի էկրանի պատկերը ավելի լավ տեսք ունի քան համակարգչի էկրանին:

Ունեցած գույների պակասը փոխհատուցելու ուղղիներից մեկը դա համակարգչային պատկերի կեղծ գուներանգումն է: Գոյություն ունեն կեղծ գուներանգման շատ տարբերակներ բայց նրանք բոլորը հիմնված են նույն սկզբունքի վրա:

Գաղափարը հետևյալն է ª փոխարինել ներկապնակում բացակայող գույներով պիկսելները ներապնակի գույներով ուրվանկարներով: Կեղծ գուներանգումը հիմնվծ է այն բանի վրա որ մարդու աչքը խառնում է երկու իրար կողք գտնվող պիկսելների գույները , ընկալելով այն որպես մի երրորդ գույն:

* 1. **Ռաստերային պատկերները և նրանց հիմնական բնութագրությունները**

Պատկերները, որոնք նախատեսված են համակարգչային մշակման համար, տարբերվում են ավանդական մեթոդներով ստեղծված պատկերներից: Այդ առանձնահատկությունը ի հայտ է գալիս այն պատճառով, որովհետև համակարգչային գրաֆիկան ունի երկու ձև ` ռաստերային և վեկտորային:

Քանի որ համակարգիչը հաշվողական մեքենա է, որը մշակում է թվեր, պատկերներ, ապա մինչ դեռ այն դառնա հասկանալի համակարգչին, անհրաժեշտ է նկարագրել այն մաթեմատիկորեն կամ թվայնացնել: Գոյություն ունեն մաթեմատիկական բնութագրման երկու ձև ` անալիտիկ ` մաթեմատիկական բանաձևերի օգնությամբ և թվերի զանգվածների նկարագրություն: Ռաստերային կամ բիտային ձևը համակարգչային գրաֆիկայում իր գոյության համար պարտական է պատկերների թվայնացման երկրորդ ձին: Ենթադրենք մենք ունենք որևէ պատկեր: Եթե նրա մակերեսի վրա տանենք ուղղահայաց և հորիզոնական գծեր, որոնք տեղադրված կլինեն միմյանցից հավասար հեռավորությունների վրա, ապա արդյունքում կստացվի ցանց, որը բաղկացած կլինի քառակուսի բջիջներից և ամեն մի քառակուսի պարունակում է պատկերի որևէ էլեմենտ: Գույնը ամեն բջիջի համար փոփխական կլինի: Կարելի է ամեն մի բջիջի համար ընտրել մի ինչ որ հաստատուն արժեք գունային երանգի համար, օրինակ հասարակ փոքրացման մեթոդով: Եթե այժմ համարակալենք մեր բջիջները սկսած սկզբից մինչև վերջ, մենք կաստանանք զույգ թվերի ամբողջություն, որոնցից առաջինը իրենից ներկայացնում է քառակուսու համարը, իսկ երկրորդը բնութագրում է գունային երանգի փոքրացումը: Հենց այս մեթոդն է կիրառվում ռաստերային պատկերների նկարագրության ժամանակ:

Ցանցը, որը ստեղծվել է իրական թվայնացման համար, պարունակում է մեծ քանակությամբ բջիջներ, որոնք այնքան փոքր են, որ մարդը դրանք չի տեսնում ` ընդունելով բոլոր պատկերները որպես մեկ ամբողջություն: Ցանցը ստացել է ռաստերային քարտ անվանումը, իսկ նրա միավոր էլեմենտը ` ռաստեր: Ռաստերային քարտը իրենից ներկայացնում է երեք թվերի հավաքածու (զանգված) ` հարթության վրա երկու կոորդինատները և նրա գույնը:

Ռաստերը ` պիկսելների մատրից է: Ամեն պիկսել կարող է ունենալ իր սեփական գույնը: Տարբեր գույների բազմությունը կազմում են պատկերը:

Կախված տարածությունում պիկսելի դիրքից տարբերվում են ` քառակուսային, ուղղանկյունային, վեցանկյունային և այլ տիպի ռաստրեր: Պիկսելների դիրքը նկարագրելու համար օգտագործում են տարբեր կոորդինատային համակարգեր: Բոլորի համար ընդհանուր համակարգ է հանդիսանում այն, որ պիկսելների կոորդինատները առաջացնում են արժեքների դիսկրետ շարք: Հաճախ օգտագործվում է ամբողջ թվերի կոորդինատների համակարգը, որի պիկսելների համարները սկսում են (0,0) վերևի ձախ անկյունից: Այսպիսի համակարգը մենք կօգտագործենք մեր նպատակների համար, քանի որ այն հարմար է օգտագործել գրաֆիկական արտածումների ալգորիթմների դիտարկման համար:

*Ռաստերային գրաֆիկայի առավելությունները `*

* Հիմնականը պարզությունն է, և որպես հետևանք, պատկերվող ինֆորմացիայի ավտոմատացման տեխնիկական իրականացումը: Գոյություն ունի արտաքին սարքերի զարգացած համակարգ, որը նախատեսված է նկարների, սլայդերի, լուսանկարների և այլ պատկերավոր օբյեկտերի համար: Դրանց վերաբերվում են սկաներները, տեսախցիկները, թվային լուսախցիկները, գրաֆիկական պլանշետը:
* Ֆայլերի ֆորմատները, որոնք նախատեսվախ են ռաստերային պատկերները պահելու համար և որոնք հանդիսանում են որպես ստանդարտ, այդ պատճառով կարևոր չէ թե որ գրաֆիկական խմբագրի միջոցով է ստացվել տվյալ պատկերը: Ֆայլը, որը պահպանում է ռաստերային պատկերը, հեշտությամբ բացվում է և ներմուծվում են վեկտորային և ռաստերային գրաֆիկայի խմբագրերում, ինչպես նաև այլ ծրագրերում:

*Ռաստերային գրաֆիկայի թերությունները `*

* Առաջին թերությունն այն է, որ մինչ ռաստերային պատկեր ստեղծելը անհրաժեշտ է տալ թույատվության որոշակի արժեք (պիկսելների քանակը մեկ միավոր երկարոթյան) և գույնի խորությունը, ինչպես նաև երկրաչափական չափը (պատկերի երկարությունը և լայնությունը, այսինքն մակերեսը):
* Երկրորդ թերությունը ի հայտ է գալիս այն ժամանակ, երբ մենք ցանկանում ենք տեսածրել ոչ այնքան մեծ նկարը մաքսիմալ թույատվությամբ և գույնի խորությամբ:
* Երրորդ թերությունը ի հայտ է գալիս այն ժամանակ, երբ մենք ուզւոմ ենք թեթևակի շրջել պատկերը փոքր անկյան տակ: Անմիջապես հայտնաբերվում է հստակ գծերը ձևափոխվում են հստակ ՙաստիճանների՚: Դա նշանակում է, որ ցանկացած ձևափոխության ժամանակ առանց աղավաղումենրի չի տեղի ունենում:
  1. **Գույների քանակը**

Գույների քանակը նույնպես ռաստերի շատ կարևոր բնութագրերից է:

Դասկարգենք պատկերները հետևյալ ձևով `

* երկգունանի (բինար) ` մեկ բիտ մեկ պիկսելին: Երկգունանիների մեջ միշտ հանդիպում են սև և սպիտակ պատկերներ:
* կիսերանգային ` մոխրագույն կամ այլ գույնի աստիճանավորումը: Օրինակ ` 256 գույների աստիճանավորումը (մեկ պիկսելին մեկ բայտ):
* գունավոր պատկերներ ` մեկ պիկսելին երկու բիտ և ավել: Գույնի խորությունը 16 բիտ է մեկ պիկսելին: Այն ստացել է High Color անվանումը, ինկ 24 բիտ մեկ պիկսելին (16.7 մլն գույն) ` True Color:

Դիտարկենք գույնը ալիքային հատկության տեսանկյունից: Ալիքի հատկություններից մեկը դա ալիքի երկարությունն է ` հեռավորությունը, որով անցնում է ալիքը մեկ տատանման ընդացքում: Մեներանգ կոչվում է այն ճառագայթը, որի սպեկտրը իրենից ներկայացնում է գիծ, որը համապատասխանում է ալիքի միակ երկարությանը: Բավականին որակյալ մեներանգ ճառագայթի աղբյուր է հանդիսանում լազերը ` հենց այդ պատճառով նրա ճառագայթը հեշտ է կիզակետել: Մեներանգ ճառագայթի գույնը որոշվում է ալիքի երկարությամբ: Ալիքի երկարության միջակայքը տեսանելի լույսի համար 380-400 նմ մինջև 700-780 նմ: Այս միջակայքում մարդու տեսողության զգայունությունը հաստատուն չէ: Առավել զգայունություն է նկատվում ալիքի այն երկարության դեպքում, որը համապատասխանում է կանաչ գույնին(նկար 4):

Գույնի բնաութագրման համար օգտագործվում են հետևյալ նշանակությունները`

* Գունային երանգ: Գունային երանգը թույլ է տալիս տարբերել մեկ գույնը մյուսից` օրինակ կարմիրը կանաչից, դեղինից և այլն:
* Պայծառություն: Որոշվում է էներգիայով, լուսային ճառագայթման ինտենսիվությամբ: Արտահայտում է ընկալելի գույնի քանակով:
* Հագեցվածություն կամ էլ երանգի մաքրությունը:Արտահայտվում է սպիտակ գույնի մի մասի առկայությամբ: Իդեալական մաքուր գույնում սպիտակ գույնը բացակայում է:

Այս երեք կետերը թույլ են տալիս նկարագրել բոլոր գույները և երանգները:

**1.4. BMP ֆորմատի կառուցվածքը**

BMP ֆորմատի հետ աշխատող ծրագրերը շատ են: Կարող է ունենալ հետևյալ ընդլայնումները՝  .bmp, .dib կամ .rle:

BMP ֆայլը բաղկացած է 4 մասից՝

1. Ֆայլի գլխամաս (BITMAPFILEHEADER)
2. Նկարի գլխամաս (BITMAPINFOHEADER, может отсутствовать). BITMAPV4HEADER (Win95, NT4.0) BITMAPV5HEADER (Win98/Me, 2000/XP)
3. Պալիտրա (կարող է նաև բացակայել)
4. Հենց նկարը:

Աղյուսակ 1

|  |  |  |
| --- | --- | --- |
| **Շեղում** | **Դաշտի երկարությունը** | **Դաշտի նկարագչությունը(ինչ է գտնվում այստեղ)** |
| **Ֆայլի գլխամաս** | | |
| 0 | 2 | Կոդ 4D42h - BM տառերը |
|  | 4 | Ֆայլի չափսը բայտերով |
| 6 | 2 | 0 (Չափային դաշտ) |
| 8 | 2 | 0 (Չափային դաշտ) |
| 10 | 4 | Շեղումը, որից սկսում է հենց ինքը պատկերը(ռաստր) |
| **BITMAP-ի գլխամաս (պատկերի մասին ինֆորմացիա)** | | |
| 14 | 4 | BITMAP գլխամասի չափսը(բայտերով) հավասար է 40 |
| 18 | 4 | Պատկերի լայնությունը պիկսելներով |
| 22 | 4 | Պատկերի բարձրությունը պիկսելներով |
| 26 | 2 | Հարթությունների քանակը պետք է հավասար լինի 1-ի |
| 28 | 2 | Բիտ/պիկսել: 1, 4, 8 կամ 24 |
| 30 | 4 | Սեղմման տիպը |
| 34 | 4 | 0 կամ սեզմված պատկերի չափսը բայտերով |
| 38 | 4 | Հորիզոնական , պիկսել/մ |
| 42 | 4 | Ուզզահայաց , պիկսել/մ |
| 46 | 4 | Օգտագործվող գույների քանակը |
| 50 | 4 | “Կարևոր” գույների քանակը |
| **Գունապնակ (N քաակությամբ գույների համար քարտեզ), եթե օգտագործվում է** | | |
| 54 | 4\*N | Ներկապնակ |

Ֆայլի գլխամասը սկսվում է «BM» ստորագրությամբ, որին հետևում է ֆայլի երկարությունը արտահայտված բայթերով: Հաջորդող չորս բայթերը արխիվացված են հետագա ընդլայնված ֆորմատի համար: Գլխամասը ավարտվում է ֆայլի սկզբից մինչև պատկերի մասին գրված ինֆորմացիայի տարբերությամբ:

Տվյալների գլխամասը սկսվում է սեփական երկարությունից (այն կարող է փոխվել, բայց 256 գունային ֆայլի համար 40 բայթ է) և պարունակում է պատկերի չափերը, գույնի ներկայացման բնութագրերը և այլ պարամետրեր:

Պատկերի երկարությունն ու լայնությունը տրվում են պատկերացանցի կողմից:

Գույնի խորությունը գույնի նկարագրման ամենակարևոր բնութագիրն է համարվում, և չափվում է կետում բիթերով:

Սեղմում՝ Bmp ֆայլերում սովորաբար չի օգտագործվում, բայց գլխամասում տեղ հատկացված է: Սովորաբար 0-ի է հավասար, և դա նսհանակում է, որ նկարը սեղմված չէ:

Պատկերի չափեր – հիշողության մեջ բայթերի քանակ, որոնք անհրաժեշտ են պատկերը պահելու համար, չհաշված պալիտրայի տվյալները:

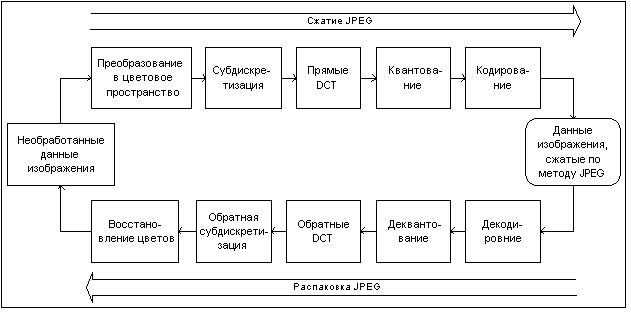
Գույների քանակը կրճատում է պալիտրայի աղուսյակի չափերը, եթե իհարկե պատկերում իսկապես կան ավելի քիչ գույներ, քան դա թույլ է տալիս գույնի ընտրված խորությունը: Սակայն գործնականում նման ֆայլեր համարյա չեն հանդիպում: Եթե գույների քանակը ընդունում է արժեք, որը հավասար է գույնի խորության թույլատրելի մաքսիմալ արժեքին, օրինակ 256 գույների ժամանակ 8 բիթի դեպքում, դաշտը զրոյացվում է:

* 1. **JPEG գրաֆիկական ձևաչափ**

JPEG-ը հանդիսանում է այն ձևաչափերից մեկը, որը օգտագործում է դիսկրետ կոսինուս ձևափոխության վրա հիմնված կոդավորման սխեման (DCT): Սեղմման այս ալգորիթմը հիմնված է պիկսելների միջև տարբերության վրա: Սովորաբար այդ սեղմումը ուղեկցվում է տվյալների որոշակի կորուսներով: JPEG-ը նպատակահարմար է բազմաերանգ (բազմամակարդակ) պատկերներում, քանի որ նրանցում հարևան պիկսելների միջև տարբերությունը աննշան է: Գործնականում JPEG-ը աշխատում է միայն այն պատկերների հետ, որոնք ունեն 4 կամ 5 բիտ/պիկսել խորության գունային կանալ: Ստանդարտային հիմունքները օգտագործում են 8 բիտ/պիկսել մուտքային նմուշի խորություն: Սակայն այս դեպքում վերջնական արդյունքը կարող է չգոհացնել, քանի որ հարևան պիկսելների միջև տարբերությունը մեծ է:

JPEG-ում սեղմման պրոցեսը բաղկացած է քայլերի հետևյալ հաջորդականությունից(նկ.1).

1. պատկերի ձևափոխումը օպտիմալ գունային միջավայր,
2. գունային բաղադրիչնեի ենթադիսկրետիզացիան` ըստ պիկսելների խմբերի,
3. դիսկրետ կոսինուս ձևափոխություն` պատկերի ավելորդ տվյալների նվազեցման համար,
4. DCT-ի գործակիցների յուրաքանչյուր բլոկի քվանտավորում` բազմաթիվ ֆունկցիաներին դիմման միջոցով,
5. արդյունքում ստացված գործակիցների կոդավորում` Հաֆֆմենի ալգորիթմի միջոցով` ավելորդ ինֆորմացիայի հեռացման նպատակով:



Նկ.1 JPEG ձևափոխության կառուցվածքը

* **Գունային միջավայր:**

JPEG - ի ալգորիթմը հնարավորություն է տալիս կոդավորել ցանկացած գունային միջավայրի պատկերներ: Այն կոդավորում է գունային մոդելի յուրաքանչյուր բաղադրիչը առանձին, ինչը ապահովում է նրա լիովին անկախությունը գունային միջավայրի ցանկացած մոդելից (օրինակ` RGB, HSI կամ CMYК մոդելից):

Գունային միջավայրին ( պայծառություն/գունային մակարդակ) դիմման դեպքում, օրինալ` ինչպես YUV կամ YcbCr, ստացվում է սեղմման լավագույն արդյունք: Y բաղադրիչը իրենից ներկայացնում է ինտենսիվությունը, իսկ U-ն և V-ն` գունային մակարդակը: Այս մոդելը կարող է փոխարինվել RGB-ով: В полутоновых պատկերներում ( մոխրագույնի հաճախականությունում) օգտագործվում է միայն Y բաղադրիչը: RGB մոդելի փոխակերպումը YcbCR-ի մոդելի իրականացվում է հետևյալ կերպ.

Y = 0,299R + 0,587G + 0,144B

Cb = -0,1687R – 0,3313G + 0,5B + 128

Cr = 0,5R – 0,4187G – 0,0813B + 128

Հետադարձ փոխակերպումը YcbCr-ից RGB-ի իրականացվում է ըստ հետևյալ հարաբերակցությունների.

R = Y + 1,402(Cr - 128)

G = Y – 0,34414(Cb - 128) – 0,71414(Cr - 128)

B = Y + 1,772(Cb – 128)

* **Գունային բաղադրիչների ենթադիսկրետիզացիան:**

Համակարգչի էկրանին մարդու անզեն աչքը երբեք չի տեսնում իրական գույներով պատկեր: Դա բացատրվում է ПЭВМ-ի հիշողությունում թվային ներկայացման սահմանափակ հնարավորություններով, մոնիտորով և վիդեոկարտայով պատկերների արտապատկերման խեղաթյուրումներով: Արդյունքում համակարգչի էկրանին արտապատկերվում է, կախված ընտրված վիդեոռեժիմից, իրականին մոտ գույներ:

JPEG ստանդարտն առաջարկում է դիսկրետիզացիայի գործակիցների որոշման մի քանի տարբեր եղանակներ կամ ենթադիսկրետիզացիայի կանալի հարաբերական արժեքներ: Պայծառության կանալ միշտ մնում է նույնությամբ (դիսկրետիզացիան` 1:1): Մյուս երկու կանալներին սովորաբար տրվում է սուբդիսկրետիզացիան 2` հորիզանական, և 1:1 կամ 2:1` ուղղահայաց ուղղությամբ:

RGB, HSI և CMY գունային մոդելները օգտագործում են օգտակար տեսողական ինֆորմացիայի բոլոր երեք բաղադրիչները: Դա նվազեցնում է տվյալների սեղմման մակարդակը: Եթե ինֆորմացիայի ներկայացումը թվային է, ապա նրանում սահմանված է ազդանշանի քվանտավորման որոշակի համակարգ: Արդյունքում քվանտավորման մակարդակը որոշվում է գույնի խորությամբ, այսինքն` կոդային կոմբինացիաների քանակով, որոնք օգտագործվում են գույնի կոդավորման համար: Այս պրոցեսը կոչվում է գույնի քվանտավորում: Եթե քվանտավորման մատրիցում բացակայում է բարձր հաճախության ինֆորմացիայի` մուտքային մատրիցին համապատասխան արժեքը, ապա հաճախ սեղմվում է մինչև իր սկզբնական չափերի կեսը կամ նույնիսկ ավելի շատ: Իրական լուսանկարչական պատկերները հաճախ անհնար է սեղմել սեղմման անկորուստ մեթոդների միջոցով, այդ պատճառով 50% սեղմումը համարվում է բավականին լավ արդյունք:

* **Պատկերի սեգմենտացիա:**

Պատկերի սեգմենտացիան կիրառվում է նրա երկու և ավելի սեգմենտների բաժանման նպատակով: Դա հեշտացնում է պատկերի տվյալների բուֆերացումը համակարգչի հիշողությունում, արագացնում է նրանց կամայական ընտրությունը սկավառակից, և թույլ է տալիս պահպանել պատկերը 64х64 Կբ-ից մեծ չափերով: JPEG-ը առաջարկում է պատկերի սեգմենտացիայի 3 տիպեր`պարզ, բրգաձև և կոմբինացված (համակցված):

Պարզ սեգմենտացիայի դեպքում պատկերը բաժանվում է երկու կամ ավելի ֆիքսված չափսերով սեգմենտների: Բոլոր պարզ սեգմենտները կոդավորվում են ձախից աջ և վերևից ներքև, հանդիսանում են հարակից և չհամընկնող: Սեգմենտները պետք է ունենան ընտրության և բաղադրիչների ինդենտիֆիկատորների միևնույն քանակը, և պետք է կոդավորված լինեն միևնույն սխեմայով: Պատկերի ներքևի և աջ մասերը կարող են լինել ավելի փոքր չափսերի, քան <<ներքին>> սեգմենտները:

Բրգաձև սեգմենտացիայի դեպքում նույնպես պատկերը բաժանվում է սեգմենտների, որոնցից յուրաքանչյուրը, իր հերթին, բաժանվում է ավելի փոքր սեգմենտների: Այս դեպքում օգտագործվում են թույլատվության տարբեր մակարդակներ: Այդպիսի պրոցեսի մոդել է հանդիսանում JPEG պատկերի սեգմենտային բուրգը (JPEG Tiled Image Pyramid, JTIP):

JTIP սխեմայում պետկերի հերթական շերտերը պահպանվում են տարբեր թույլատրություններով: Առաջին պատկերը, որը գտնվում է բուրգի վերևում, զբաղեցնում է էկրանի հաստատված չափսերի մեկ տասնվեցերորդ մասը և կոչվում է վինետկա : Այն կիրառվում է պատկերի արագ արտապատկերման համար և հատուկ նշանակություն դիտելու ծրագրերով (բրաուզերներով) աշխատանքների ժամանակ: Հաջորդ պատկերը զբաղեցնում է էկրանի մեկ քառորդ մասը և կոչվում է մանետկա: Սովորաբար դա օգտագործվում է այն դեպքերում, երբ անհրաժեշտ է էկրանի վրա ունենալ երկու և ավելի պատկերներ միաժամանակ: Այնուհետև հաջորդում են ցածր թույլատրությամբ ամբողջ էկրանով պատկերները, ապա պատկերները` ասատիճանաբար աճող թույլատրությամբ, և, վերջապես, օրիգինալ պատկերը: Բրգաձև սեգմենտացիայի դեպքում նպատակահարմար է ներքին սեգմենտացիայի պրոցեսը, երբ յուրաքանչյուր սեգմենտ կոդավորվում է որպես JPEG տվյալների մեկ հոսքի մաս:

Համակցված սեգմենտացիան թույլ է տալիս պահպանել և արտապատկերել պատկերի տարբերակները թույլատրության մի քանի մակարդակներով խճանկարի տեսքով: Համակցված սեգմենտացիան թույլ է տալիս տարբեր չափսերի համընկնող սեգմենտների ներկայությունը` տարբեր մասշտաբային գործակիցներով և սեղմման պարամետրերով: Յուրաքանչյուր սեգմենտ կոդավորվում է առանձին և կարող է համակցվել այլ սեգմենտների հետ առանց կրկնվող դիսկրետիզացիայի:

Այսպիսով, որքան բարձր է խտացման մակարդակը, այնքան մեծ է տվյալների կորուստը և փոքր է պատկերի որակը: Օգտագործելով JPEG-ը` կարելի է ստանալ 1-500 անգամ ավելի փոքր ֆայլ, քան BMP-ով: Դա կիրառվում է РС-ի և Macintosh-ի կողմից:

* **Դիսկրետ կոսինուս ձևափոխություն:**

Ալգորիթմի գլխավոր բաղադրիչ է հանդիսանում դիսկրետ կոսինուս ձևափոխությունը: Այն իրենից ներկայացնում է Ֆուրյեի ձևափոխության տարատեսակներից և նրա նման ունի հետադարձ ձևափոխություն:Գրաֆիկական պատկերները կարելի է դիտարկել որպես տարածական ալիքների համակցություն, որտեղ X և Y առանցքները համապատասխանում են նկարի երկարությանն ու լայնությանը, իսկ Z առանցքը` նկարի համապատասխան պիկսելի գույնին: Դիսկրետ կոսինուս ձևափոխությունը թույլ է տալիս կատարել անցում նկարի տարածական ներկայացումից սպեկտրալ ներկայացման և հակառակը: Դիսկրետ կոսինուս ձևափոխությունը պիկսելների NxN չափանի մատրիցը համապատասխան չափի հաճախականային գործակիցների մատրիցով: Ուղիղ և հակադարձ դիսկրետ կոսինուս ձևափոխության ֆորմուլան հետևյալն է.

* Դիսկրետ կուսինուս ձևափոխության ֆորմուլա (ДКП)

* Հակադարձ դիսկրետ կուսինուս ձևափոխության ֆորմուլա (ДКП)

Ստացված գործակիցների մատրիցում ցածրհաճախականային բաղադրիչները գտնվում են վերևի ձախ անկյանը մոտ, իսկ բարձրհաճախականայինները` ներքևի աջ անկյանը: Դրա պատճառն այն է, որ համակարգչի էկրանին գրաֆիկական պատկերների մեծամասնությունը կազմված է ցածրհաճախականային ինֆորմացիայից: Այսպիսով` դիսկրետ կուսինուս ձևափոխությունը հնարավորություն է տալիս որոշել, թե ինֆորմացիայի որ մասն է կարելի հեռացնել այնպես անվնաս, որ նկարը չենթարկվի լուրջ աղավաղումների:

Դիսկրետ կուսինուս ձևափոխության արդյունքում սեղմում և կորուստներ տեղի չեն ունենում, սակայն պատրաստվում է ինֆորմացիա կորուստներով սեղմման կամ կլորացման համար:

Դիսկրետ կուսինուս ձևափոխության ժամանակ մատրիցի յուրաքանչյուր տարրի արժեքը կախված է պատկերի չափերից: Քանի որ օգտագործվում են երկու ցիկլեր, ապա հաշվարկի ողջ ժամանակահատվածը կլինի O(NxN), և գործնականորեն անհնար կլինի իրականացնել ամբողջ պատկերի ձևափոխությունը միանգամից: Այդ պատճառով պատկերը բաժանվում է 8x8 չափանի հատվածների:

Դիսկրետ կոսինուս ձևափոխությունն իրականցվում է երկու ներդրված ցիկլերի միջոցով: Ցիկլի մարմինը կատարվում է NxN անգամ մատրիցի յուրաքանչյուր տարրի համար: Գործակիցների հաշվարկման ավելի էֆեկտիվ տարբերակ է մատրիցների բազմապատկումը:

*ДКП = КП \* Точки \* КПт*

Որտեղ`

ДКП – դիսկրետ կոսինուս ձևափոխություն

КП – կոսինուս ձևաթոխության NxN չափանի մատրից, որի տարրերը հաշվարկվում են հետևյալ կերպ`

*Точки* – NxN չափանի մատրից, որը կազմված է պատկերի

Кпт - վերադասավորված КП մատրից

Այս դեպքում հաշվարկի ժամանակը 2xN է:

Գործակիցների կլորացումը: Այն իրենից ներկայացնում է բիտերի քանակի կրճատման այնպիսի պրոցես, որի արդյունքում դիսկրետ կոսինուս ձևափոխության գործակիցների մատրիցը պահպանվում է առանց կորուստների: Դա իրականացվում է կլորացման մատրիցի միջոցով: Դիսկրետ կոսինուս ձևափոխության մատրիցի յուրաքանչյուր տարրին համապատասխանում է կլորացման մատրիցի մեկ տարր: Վերջնական մատրիցը ստացվում է դիսկրետ կոսինուս ձևափոխության և կլորացման մատրիցների տարրի հարաբերության` դեպի մոտակա ամբողջ թիվը կլորացումից: Սովորաբար վերջին մատրիցի տարրերի արժեքները աճում են ձախից աջ և վերևից ներքև:

Կլորացման մատրիցի ընտրությունը: JPEG-ը կիրառում է ISO-ի կլորացման մատրիցը: Կլորացման մատրիցը ստեղծվում է հետևյալ պարզ ալգորիթմի հիման վրա: Որպեսզի որոշվի կլորացման մատրիցում աճման արժեքը, [1,25] միջակայքից ընտրվում է մեկ արժեք, որը կոչվում է որակի գործակից: Այնուհետև մատևիցը լրացվում է հետևյալ սկզբունքով.

for (i = 0; i < N; i++)

for (j = 0; j < N; j++)

Matrix[i][j] = 1 + (1 + i + j) \* QualityFactor;

Արդյունքում ստացվում է տվյալ տարրի անկյունագծային հաջորդ տարրը: Այդպիսի օրինակ դիտարկված է հետևյալ աղյուսակում (Աղյուսակ 2)

Աղյուսակ 2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 |
| 5 | 7 | 11 | 13 | 15 | 17 | 19 | 21 |
| 7 | 11 | 13 | 15 | 17 | 19 | 21 | 23 |
| 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 |
| 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 |
| 13 | 15 | 17 | 19 | 21 | 23 | 25 | 27 |
| 15 | 17 | 19 | 21 | 23 | 25 | 27 | 29 |
| 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 |

Կարևոր է նշել, որ գործակիցների կլորացման մատրիցի նմանատիպ արժեքների դեպքում դիսկրետ կոսինուս ձևափոխության մատրիցի (7,7) տարրը չպետք է փոքր լինի 16-ից, որպեսզի կլորացումից հետո ստացվի 0-ից տարբեր արժեք: Սա այն միակ փուլն է, որի արդյունքում տեղի է ունենում ինֆորմացիայի կորուստ:

Այս փուլի իրականացումից հետո, փաստորեն, առանձնացվում է բարձրհաճախականային և ցածրհաճախականային ինֆորմացիան, որոնցից կազմված է պատկերը, որից հետո հեռացվում է բարձրհաճախականային ինֆորմացիան: Մինչ այդ կոմպրեսորը բաժանում է DCT-ի յուրաքանչյուր ելքային արժեք <<քվանտավորման գործակցի>> վրա` կլորացնելով արդյունքը մինչև ամբողջ թիվ: Որքան մեծ է քվանտավորման գործակիցը, այնքան մեծ է տվյալների կորուստը: DCT-ի ելքային բլոկի 64 կարգերից յուրաքանչյուրն ունի համապատասխան քվանտավորման գործակից: Պայծառության և գույնի տվյալները ունեն առանձին քվանտավորման աղյուսակներ: Ընդ որում, գունայիբ տվյալների քվանտավորման գործակիցներն ունեն ավելի մեծ արժեքներ: JPEG-ն օգտագործում է պատրաստի քվանտավորման աղյուսակներ, որոնք գրանցվում են սեղմված ֆայլում, որպեսզի դեկոմպրեսորն իմանա, թե ինչպես վերականգնի DCT գործակիցները(մոտավորապես):

* **Կոդավորում:**

JPEGկոդավորիչի աշխատանքի վերջնական փուլը կոդավորումն է, որն իրենից ներկայացնում է երեք գործողություններ դիսկրետ կոսինուս ձևափոխության մատրիցի շուրջ` սեղմման մակարդակը բարձրացնելու նկատմամբ:

Առաջին գործողությունը մատրիցի (0,0) կարգում գտնվող գործակցի բացարձակ արժեքի փոխարինումն է բացասականով: Քանի որ պատկերի հարևան բլոկները նման են միմյանց, ապա հերթական (0,0) տարրի կոդավորումը նախորդի համեմատ չի տալիս էական փոփոխություններ: Դիսկրետ կոսինուս ձևափոխության մատրիցի գործակիցները դասավորված են զիգզագաձև: Զրոյական արժեքները կոդավորման կրկնման ալգորիթմով (RLE) կոդավորելուց հետո արդյունքը մշակվում է <<էնտրոպիայի կոդավորմամբ>>, այսինքն` Հաֆֆմանի ալգորիթմով կամ թվաբանական կոդավորմամբ:

<<Էնտրոպիայի կոդավորում>>: Որպես կոդավորման պարզեցված սխեմայի արդյունք են հանդիսանում հետևյալ երեք տեսքերը. զրոների քանակ, բիտերի քանակ, գործակից, որտեղ

1. զրոների քանակ - դիսկրետ կոսինուս ձևափոխության մատրիցում տվյալ ոչ զրոյական տարրին նախորդող զրոների քանակն է,
2. բիտերի քանակ – գործակցի արժեքը կոդավորող հաջորդող բիտերի քանակը,
3. գործակից- դիսկրետ կոսինուս ձևափոխության մատրիցի ոչ զրոյական տարրի արժեքը:

Բիտերի քանակի և գործակցի կապը բերված է հետևյալ աղյուսակում (Աղ. 3).

Աղյուսակ 3

|  |  |
| --- | --- |
| Բիտերի քանակը | Գործակից |
| 1 | [-1, 1] |
| 2 | [-3, -2], [2, 3] |
| 3 | [-7, -4], [4, 7] |
| 4 | [-15, -8], [8, 15] |
| 5 | [-31, -16], [16, 31] |
| 6 | [-63, -32], [32, 63] |
| 7 | [-127, -64], [64, 127] |
| 8 | [-255, -128], [128, 255] |
| 9 | [-511, -256], [256, 511] |
| 10 | [-1023, -512], [512, 1023] |

Այս կոդավորումը չնայած Հաֆֆմանի կոդավորումից էֆեկտիվ չէ, բայց որոշ տվյալների համար այն տալիս է տրամաբանական արդյունքներ:

Այս փուլի ավարտից հետո JPEG տվյալների հոսքը պատրաստ է հաղորդակցման ուղիներով փոխանցման կամ ֆայլի ֆորմատի մեջ ինկապսուլիացիայի:

* 1. **AVI ֆորմատի կառուցվածքը**

Avi (Audio Video Interleave) – ը առաջին անգամ օգտագործվել է Microsoft-ի կողմից 1992թ.  Video for Windows փաթեթում:

Avi ընդլայնումով ֆայլերը կարող են պարունակել աուդիո և վիդեո տվյալներ, որոնք սեղմված են տարբեր կոդեկների կոմբինացիաներով, ինչն էլ թույլ է տալիս վիդեոն սինխրոն կերպով վերարտադրել աուդիոյի հետ: Avi ֆայլը, կախված թե ինչ կոդեկով է կոդավորվել կամ վերծանվել, կարող է պարունակել տարբեր տեսակի սեղմված տվյալներ (օրնակ՝ DivX-վիդեո + WMA-աուդիո կամ Indeo - վիդեո + PCM - աուդիո):  Ինչպես և DVD-ն, այնպես էլ AVI-ի ֆայլերը պարունակում են բազմահոսք աուդիո և վիդեո:

Բոլոր Avi ֆայլերը պարտադիր պարունակում են 2 <<LIST chunk'а>>, որոնք էլ որոշում են ֆորմատը և տվյալների հոսքը: Avi ֆայլերը կարող են նաև պարունակել <<chunk>> ինդեքս: Այդ հավելյալ <<chunk>> ինդեքսը որոշում է վիդեոտվյալների տեղը հոսքում:

Avi ֆայլի տիպիկ կառուցվածքը հետևյալն է՝

**RIFF** ('AVI ' **LIST** ('hdrl' <заголовок>) **LIST** ('movi' <видео>) ['idx1' <индекс>])

<<chunk>>- ի տիպիկ կառուցվածքը՝

**LIST** ('hrdl' 'avih' (<заголовок AVI файла>) **LIST** ('strl' <заголовок потока 1>) **LIST** ('strl' <заголовок потока 2>) **LIST** ('odml' <расширенный заголовок AVI файла>))

Ֆայլը սկսվում է հիմնական գլխամասով: Avi ֆայլերում գլխամասը որոշվում է <<chunk>>-ով, FOURCC 'avih' հետ: Գլխամասը պարունակում է ամբողջ ֆայլի մասին հիմնական տվյալները, այնպիսին, ինչպիսիք են հոսքերի քանակը ֆայլում, վիդեոհոսքի լայնությունն ու երկարությունը և այլն:

Չսեղմված Avi ֆայլը իրենից ներկայացնում է bmp նկարների հայորդականություն: Այդ պատճառով Avi ֆայլի մեջ ինֆորմացայի թաքնագրումը վերածվում է bmp նկարների մեջ ինֆորմացիայի թաքնագրման:

**Գլուխ 2. Խնդրի դրվածքը և կատարված աշխատանքի ալգորիթմի նկարագրությունը**

**2.1 Խնդրի դրվածք**

Մարդն իր ամբողջ պատմության ընթացքում փորձել է հեշտացնել իր առորյան: Մինչև այժմ էլ մարդկությունը շարունակում է այդ ուղղությամբ և դա է պատճառը տնտեսական և ինֆորմացիոն զարգացմանը: Վերջին 50 տարվա ընթացքում, երբ ստեղծվեցին համակարգչային տեխնոլոգիաները, մարդու զարգացման տեմպերը կտրուկ աճեցին: Ըստ Մուրի օրենքի միավոր ցավալի վրա տրանզիստորների քանակը 2 անգամ շատանում է 18 ամիսը մեկ, որն իր հետևից առաջ է բերում համակարգչի կոմպոնենտների փոքրացմանը և արագագործության ու հզորության մեծացմանը: Համակարգչային տեխնոլոգիաների ստեղծման ակունքներում համակարգիչները կարելի էր համեմատել այժմյան հաշվիչների հետ, նրանք գրավում էին շատ մեծ տարածք և կարողանում էին անել միայն տարրական գործողություններ: Այն ժամանակ մարդիկ երևակայել անգամ չէին կարող թե ինչքան կփոխվի կյանքը համակարգիչների շնորհիվ:

70-ական թվականների կեսերից մարդիկ սկսեցին մտածել համակարգիչների օգտագործման մասին գրաֆիկական պատկերների մշակման և արտապատկերման համար: Այժմ արդեն հնարավոր դարձավ համակարգչով մշակել մեծ չափի նկարներ և նույնիսկ վիդեոներ: Ստեղծվեցին նկարները մշակող, որակը բարձրացնող, գրաֆիկական և ֆոտո էֆեկտներ ավելացնող ծրագրեր, ինչպես նաև վիդեո ֆայլերը մոնտաժ անող, էֆեկտներ ավելացնող ծրագրեր:

Այժմ համակարգչային տեխնիկան այնքան է զարգացել որ հնարավոր է դարձել վիդեո հոսքի ռեալ-ժամանակում մշակումը: Դա բազմաթիվ նոր հնարավորություններ է ստեղծում ինտելեկտուալ և ավտոմատացված համակարգեր ստեղծելու համար: Ի՞նչ է ավտոմատացումը: Ավտոմատացումը դա տվյալ խնդրի լուծման համար մարդու կատարած գործողություններ հերդականության, ալգորիթմների մշակումը և դրանց կիրառումն է ավտոմատ համակարգերի մեջ, վորոնցից են օրինակ համակարգիչները, ռոբոտները և այլ տեսակի խելացի սարքավորումները: Ավտոմատացումը շատ անգամ մեծացնում է գործողությունների կատարման և խնդրի լուծման արագությունը, հեշտացնում և քչացնում է մարդու աշխատանքը, մեծացնում է խնդրի կատարման ծավալները, կազմակերպում է մասսայական արտադրություն և խնդիրների լուծման զուգահեռություն և ամենակարևորը արժեքի իջեցում:

Գործնականում կարելի է ավտոմատացնել շատ խնդիրներ, հենց այդ գործողությամբել մենք կզբաղվենք:

Խնդիրը կայանում է նրանում, որ ստացիոնար վիդեո տեսախցիկով դիտվող տարածքում կատարվի օբյեկտների հայտնաբերում և հետագա գործողությունների իրականացում հայտնաբերված օբյեկտների հետ:

Տվյալ խնդիրը բացատրելու համար սկզբում հասկանանք մի քանի ֆունդամենտալ գաղափարներ և մտցնենք ֆունդամենտալ հասկացությունների բացատրություն:

* Ո՞րն է ստացիոնար տեսախցիկը:

Ստացիոնար տեսախցիկը դա տեսախցիկ է որը տեղադրված է դիտվող տարածքի նկատմամբ միշտ մի դիրքում: Այսինքն հետազոտվող տարածքը մեկ տեսախցիկի միջոցով միշտ դիտարկվում է մեկ դիրքում և այդ դիրքը չի կարող փոփոխվել որևիցե պատճառով դիտարկման և հետազոտման ընթացքում: Սակայն չի արգելվում մի քանի տեսախցիկների զուգահեռ կամ անկախ օգտագործումը, կամ թե տեսախցիկի դիրքի փոփոխումը ոչ աշխատանքային վիճակում գտնվելու ժամանակ և անպայման հետագայում նոր դիրքի համար կարգավորումների իրականացումը:

Տեսախցիկները օժտված են բազմաթիվ նկարագրող հատկանիշներով, որոնցից ամենակարևորներից են՝

* + Մեկ վարկյանի ընթացքում տեսախցիկի նկարած կադրերի (Frame) քանակը:
  + Տեսախցիկի թույլատրությունը, որը բնորոշվում է տեսախցիկով նկարված կադրերի մեջ պիկսելների քանակով:
  + Դիտման անկյունը, որը բնութագրում է տեսախցիկի նկատմամբ դիտվող տարածքի անկյունային ընդգրկման մեծությունը:
  + Եվ այլն
* Ո՞րն է խնդրի պայմաններում դիտարկվող հսկման տարածքը:

Խնդրի պայմաններում դիտարկվող հսկման տարածքը դա այն ճանապարհահատվածն է որը դիտվում է ստացիոնար կամերայի միջոցով: Այդպիսի տարածք կարող է հանդիսանալ կամայական ճանապարհահատված՝ լինի դա արագնթաց մայրուղի, քաղաքի մեջ գտնվող ցանկացած փողոց՝ անկախ նրանից լայն է այն թե նեղ, քանի երթևեկության գիծ է պարունակում, միակողմանի է թե երկկողմանի, ուղիղ է թե գալարաձև և այլն, գյուղի փողոց, կամուրջ և այլն:

Հսկման տարածքը կարող է բնորոշվել բազմաթիվ յուրահատուկ հատկանիշներով որոնցից են օրինակ՝ ճանապարհահատվածի լուսավորությունը, տարվա եղանակների պատճառով առաջացած խնդիրները, եղանակային երևույթները (անձրև, ձյուն, կարկուտ և այլն):

* Ո՞րն է դիտարկմանը ենթակա օբյեկտը:

Դիտարկման ենթակա օբյեկտ է հանդիսանում ցացկացած շարժվող օբյեկտ, որը կարող է հայտնվել դիտարկվող տարածքում: Այսպիսով օբյեկտներ կարող են հանդիսանալ մեքենաները, մոտոսիկլետները, բեռնատար ավտոմեքենաները, հեծամիվները և այլն և նուսնիսկ ուղակի քայլող մարդիկ՝ սակայն տվյալ խնդրի պայմաններում մարդիկ չեն դիտարկվելու որպես օբյեկտներ:

Օբյեկտները նույնպես բնութագրվում են բազմաթիվ հատկանիշներով: Նշենք դրանցից ամենահիմնականներն ու կարևորները խնդրի լուծման պարագայում:

* + Օբյեկտի չափերը՝ ամենաբնորոշիչ հատկանիշներից մեկն է որը թույլ է տալիս նախնական ենթադրություններ անել օբյեկտի մասին:
  + Օբյեկտի գույնը՝ գույնը նույնպես կարևոր բնութագրիչ հատկանիշ է:

Օբյեկտը կարող է լինել և ինչպես միագույն, այնպեսել կազմված լինել բազմաթիվ գույներից և գունային երանգներից, որոնց համադրությունը կազմում է օբյեկտին բնորոշ տեկստուրան:

Ըստ գունային բաղադրիչի տարբեր տեսակի օբյեկտներ կարող են բնութագրվել մի գույնով, և ընդհակառակը՝ միատեսակ օբյեկտները կարող են բնութագրվել շատ տարբեր գույներով:

Գույնի միջոցով է հիմնականում կատարվում օբյեկտների ուսումնասիրումը, որոնցից է որինակ օբյեկտների առանձնացումը ֆոնից:

* + Օբյեկտի ուրվագիծը՝ նույնպես բնութագրիչ հատկանիշ է, որը թույլ է տալիս իրականացնել օբյեկտների տեսակավորում և որևէ մի դասին պատկանելիության որոշում:
  + Օբյեկտի վիճակը՝ բնութագրիչ հատկանիշ է, որը ցույց տալիս արդյոք օբյեկտը գտնվում է կանգնած վիճակում, շարժվում է թե կատարել է ժամանակավոր կանգառ:
  + Օբյեկտի շարժման ուղղությունը և արագությունը՝ բնութագրիչ հատկանիշ է, որը օգնում է օբյեկտների հետագա ուսումնասիրությանը հայտնաբերումից հետո:
* Ո՞րն է դիտարկմանը ենթակա օբյեկտների հայտնաբերումը:

Դիտարկմանը ենթակա օբյեկտների հայտնաբերումը դա այս համակարգի հիմնական գործնթացներից մեկն է:

Համակարգի նպատակն է ուսումնասիրել թիրախաէին օբյեկտների խումբ և դրա համար առաջին հերթին պետք է հայտնաբերել այդ օբյեկտները տեսախցիկից եկած ինֆորմացիայի մեջ: Օբյեկտների հայտնաբերումը կատարվում է բազմաթիվ եղանակներով որոնք կքննարկվեն խնդրի լուծման ալգորիթմների բաժնում, միայն նշենք որ օբյեկտների հայտնաբերման գործնթացը դա հենց օբեկտների ֆոնային պատկերից աոանձնացնելու գործնթաձն է:

* Ո՞րն է դիտարկմանը ենթակա օբյեկտների հաշվարկը:

Դիտարկմանը ենթակա օբյեկտների հաշվարկը դա այս համակարգի երկրորդ հիմնական գործնթացն է:

Օբյեկտների հաշվարկը իրենից ներկայացնում է գործողություն, որի արդյունքում հաշվարկվում են ամբողջ դիտարկման ընթացքում դիտարկվող տարածքում հայտնված թիրախային օբյեկտների քանակը: Օբյեկտների քանակի հաշվարկը կարող է իրականացվել ոչ միայն բոլոր օբյեկտների համար, այլ նաև կոնկրետ դասին կամ տեսակին պատկանող թիրախային օբյեկտների համար:

Վերը նշվեցին բոլոր ֆունդամենտալ գաղափարները և մտցվեցին ֆունդամենտալ հասկացությունների բացատրություններ տվյալ խնդիրը բացատրելու և հասկանալու համար:

Այսպիսով մեր խնդիրը կայանում է նրանում, որպեսզի ձևավորվեն և մշակվեն այնպիսի ալգորիթմներ, որոնք կավտոմատեցնեն տեսախցիկով դիտվող ճանապարհահատվածում թիրախային օբյեկտների հայտնաբերման և հաշվարկման գործնթացը:

Այդ ամենի համար պետք է գրվի համակարգչային ծրագիր:

**2.2 Ալգորիթմի նկարագրությունը**

Ծրագրի աշխատանքը բնութագրող հիմնական բլոկ-սխեման պատկերված է նկար 2.2.1-ում:



նկար 2.2.1

Եվս մեկ անգամ հիշենք դիպլոմային աշխատանքի թեման, այն է՝ տեսախցիկով դիտվող ճանապարհահատվածով անցնող օբյեկտների հայտնաբերման և հաշվարկման համակարգ: Թեմայի վեռնագրից հետևում է, որ ծրագիրը պետք է բաղկացած լինի 2 հիմնական գործողություններից՝

* Օբյեկտների հայտնաբերում
* Օբյեկտների քանակի հաշվարկ

Այստեղից հետևում է, որ ծրագրի մեջ առկա է այդ 2 խնդիրների ալգորիղմային

լուծում: Ավելի մանրամասն քննարկենխ 2 խնդիրներից յուրաքանչյուրը:

Օբյեկտների հայտնաբերման համար ծրագրում մշակված և իրագործված է գործողությունների հերթականություն որի արդյունքը նպաստում է առաջին խնդրի լուծմանը: Թվարկենք այդ գործողությունների հերթականությունը՝

1. Օբյեկտների հայտնաբերման համար հետաքրքրող լոկալ տարածքի սահմանում գծանշմամբ: (Ոչ պարտադիր քայլ է, սակայն կատարման դեպքում բավականին կրճատում է ալգորիթմի կատարման ժամանակը)
2. Դիտարկվող տեսախցիկներից ստանալ տեսանյութի հարմար կադր (Frames), որը հետագայում կոգտագործվի որպես ֆոնային պատկեր (Background): (Ոչ պարտադիր քայլ է, սակայն կատարման դեպքում բարձրացնում է օբյեկտների հայտնաբերման որակը)
3. Դիտարկվող տեսախցիկներից ստանալ տեսանյութի 2 իրար հաջորդող կադրերը (Frames):
4. Նախորդ 2 կետերի կատարման արդյունքում ստացված պատկերների միջև կատարել ալգորիթմական գործողություն, որի արդյունքում ստանալ բինար պատկեր, որում սև հատվածները կպատկանեն ֆոնին (Background), իսկ սպիտակ հատվածները օբյեկտներին:
5. Նախորդ քայլում ստացված բինար պատկերում թիրախային օբյեկտների որոնում և օբյեկտների առանձնացում: Այս քայլը կատարվում է ամբողջ բինար պատկերի վրա եթե չի կատարվել առաջին քայլը և լոկալ որոնման տարածքում եթե կատարվել է առաջին քայլը: (2-րդ տարբերակը բավականին արագացնում է ալգորիթմի կատարումը)
6. Հայտնաբերված թիրախային օբյեկտների տեսակավորում և դասակարվում:
7. Դասակարգված օբյեկտների եզրագծերի հաշվում, որի արդյունքում գտնվում են նաև երկրորդական բնութագրիչ հատկություններ՝ օբյեկտների մակերես, բարձրություն, լայնություն, կենտրոնի կորդինատներ և այլն:
8. Օբյեկտների եզրագծերը ամբողջովին ընդգրկող մինիմալ մակերես ունեցող ուղղանկյունների հաշվարկ և արտապատկերում:
9. Օբյեկտների քանակի հաշվարկ:

Օբյեկտների հայտնաբերման համար օգտագործվում է գունային շեմի

Գաղափարը, որի միջոցով ստեղծվում է բինար պատկերը: Տվյալ ծրագրում գունային շեմը տրվում է Threshold Value TrackBar-ի միջոցով, որը գտնվում է Options պատուհանում: Ինչպես երևում է նկար 2.2.2-ում պատկերված բանաձևում բինար պատկերի յուրաքանչյուր կետ (Pixel) ստանալու համար մուտքային պատկերի յուրաքանչյուր կետ համեմատվում է գունային շեմի արժեքի հետ և եթե այն մեծ է գունային շեմի արժեքից ապա բինար պատկերում ունենում է նախորոք սահմանված մաքսիմալ արժեք, իսկ եթե փոքր է՝ ապա ունենում է 0-ին հավասար արժեք:



նկար 2.2.2

Գունային շեմի հետ համեմատման արդյունքով ստացված արժեքները և սկզբնական արժեքները, ինչպես նաև գունային շեմի արժեքը պատկերված են նկար 2.2.3-ում պատկերված գրաֆիկում:



նկար 2.2.3

Վերը բացատրված մեթոդը ծրագրում օգտագործվում է միքանի տարբեր դեպքերում՝

* 2 հաջորդող կադրերի հանումից ստացված պատկերի համար
* Վերջին ստացված կադրից ֆոնային կադրի հանումից ստացված պատկերի համար

Այնուհետև այդ բինար պատկերների միջև կատարվում են բիտային

համեմատման և տրամաբանական գործողություններ, վերջնական բինար նկարի ճշգրտությունը բարձրացնելու նպատակով:

Բինար պատկերի ամբողծ մակերեսում կամ կոնկրետ լոկալ տարածքում օբյեկտների դասակարգման, առանձնացման, եզրագծերի ստագման և սահմանապակող մինիմալ ուղղանկյան կորդինատների հաշվման համար օգտագործվում է հատուկ ալգորիթմ: Բացատրենք այդ ալգորիթմի աշխատանքը:

Սկզբում ընտրվում է բինար պատկերի վրա հաշվարկման սկզբնակետ, որը կամ (0. 0) կետն է կամ էլ առաջին լոկալ տարածքի սկզբնակետը: Այնուհետև կետ առ կետ ցիկլով շարժվում ենք դեպի աջ այնքան ժամանակ մինչև չհասնենք այնպիսի կետի՝ որի գունային արժեքը հավասար է 255 այսինքն սպիտակ գույնի է: Դա մեր որևէ օբյեկտի առաջին կետն է: Այդ կետի կորդինատները հիշվում են հետագայում այդ կետից շարունակելու համար:

Այնուհետև ռեկուրսիվ ալգորիթմը այդ կետից սկսած փնտրում է մոտակա հարևան կետերի մեջ սպիտակ գույն ունեցող կետը և այդպես շարունակ մինչև, որ ամբողջ օբյեկտը չի հայտնաբերվում: Որոնման ալգորիթմը կարող է աշխատել 2 ռեժիմում՝

* 4 հարևան կետերում կատարել որոնումը
* 8 հարևան կետերում կատարել որոնումը

Դա հանդիսանում է ֆունկցիայի մուտքային տվյալ:

Բացի դրանից ալգորիթմը օբյեկտի որոնման ընթացքում գունախում է օբյեկտին

պատկանող կետերը: Դա թույլ է տալիս միուս օբյեկտների որոնման ժամանակ արդեն հայտնաբերված օբյեկտների կետերը նորից հաշվի չառնել:

Այնուհետև հայտնաբերված օբյեկտի համար հաշվվում է մինիմալ շրջապատող ուղղանկյան կորդինատներն ու չափերը:

Այս բոլոր գործողություններից հետո ալգորիթմը վերադառնում է հիշված կետին և ամեն ինչ կրկնվում է նորից մինչև բոլոր օբյեկտների հայտնաբերումը և բինար պատկերի կամ լոկալ տարածքի ավարտը:

Նշենք որ ամեն օբյեկտի գույնի փոփոխման հետևանքով կատարվում է օբյեկտների առանձնացում և հետագայում կոնկրետ օբյեկտը միշտ կարելի է միանշանակորեն հայտնաբերել պատկերում:

Ալգորիթմի աշխատանքի ժամանակ կատարվում է նաև օբյեկտների դասակարգում ըստ նախորոք սահմանված չափսի և այնուհետև ամեն դասին պատկանող օբյեկտների քանակի հաշվարկ:

**Գլուխ 3. Ծրագրի աշխատանքը**

Ծրագրի թողարկումից հետո բացվում է հետևյալ պատուհանը՝

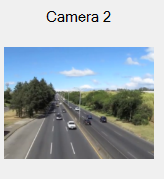
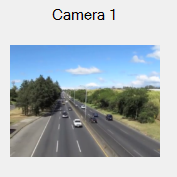


նկար 3.1

Նկար 3.1-ում պատկերված է “Costa Rica”-ի մայրուղիներից մեկը:

Այս պատուհանը հանդիսանում է գլխավոր պատուհան օգտագործողի համար:

Պատուհանի ձախ կողմում պատկերված են 4 ImageBox-եր համապատասխանաբար խաչմերուկի 4 տեսախցիկների համար (նկար 3.2): Կամ եթե մայրուղի է և տեղադրված են միքանի տեսախցիկներ տարբեր ուղղությունները նկարահանելու համար այդ տեսախցիկների տվյալները կերեվան այդ ձախ կողմի ImageBox-երում:



նկար 3.2

Ձախ կողմի ImageBox-երից ցանկացածի վրա մկնիկով սեղմելուց հետո այդ ImageBox-ին համապատասխանող տեսախցիկի տվյալները կերեվան նաև գլխավոր ImageBox-ի վրա:

Գլխավոր ImageBox-ը նախատեսված է տվյալ պահին հետևվող տեսախցիկի տվյալները ավելի մանրամասն ուսումնասիրելու համար: Այն ունի բավականին մեծ չափսեր և հարմար է Real-Time-ում տեսախցիկին հետևելու համար:

Պատուհանի վերևի հատվածում երևում է դիտվող ճանապարհահատվածի անվանումը, որպեսզի օգտագործողը միշտ տեղյակ լինի թե որ հատվածն է տվյալ պահին ուսումնասիրում: Դա կարող է լինել այն ժամանակ երբ համակարգը օգտագործվի մեծ քանակուղյամբ ճանապարհահատվածներ (մայրուղիներ, փողոցներ և խաչմերուկներ) ուսումնասիրելու համար:



նկար 3.3

Պատուհանի աջ մասում տեղադրված են 3 հիմնական Button-ներ: Համապատասխանաբար վերևից ներքև՝

* Options (նկար 3.4)
* Start (նկար 3.5)
* Stop (նկար 3.6)

Options կոճակը (նկար 3.4) սեղմելուց բացվում է պատուհան որտեղ

օգտագործողը կարող է ընտրություն կատարել բազմաթիվ օրենքների եվ կարգավորումների միջև:



նկար 3.4

Start կոճակը (նկար 3.5) սեղմելուց բոլոր ImageBox-երում սկսվում է ցուցադրվել համապատասխան տեսախցիկից ստացվող տեսանյութի տվեալները:



նկար 3.5

Stop կոճակը (նկար 3.6) սեղմելուց բոլոր ImageBox-երում ցուցադրվող տեսանյութի տվեալները կանգնեցվում են և ցուցադրվում է վերջին կադրը:



նկար 3.6

Պատուհանի աջ մասում նաև կա տեղեկատվություն (նկար 3.7) թե քանի օբյեկտ է անցել օբյեկտների քանակի հաշվման բաժանարար գիծը: Տվյալ խնդրի բաեմաններում առայժմ առանձնացվում են միայն 2 տեսակի օբյեկտ՝ մոտոցիկլետ և ավտոմեքենա:



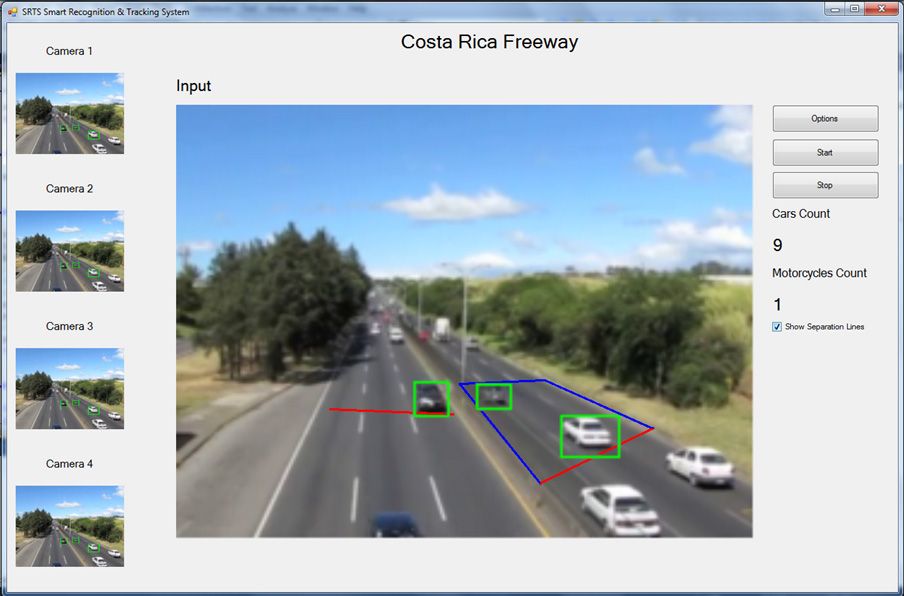
նկար 3.7

Պատուհանի աջ մասի ամենաներքևում տեղադրված է CheckBox որի միջոցով ընտրվում է արդյոք գլխավոր ImageBox-ի վրա ցույց տալ նաև բաժանարար գծերը, թե ոչ:

Երբ Show Separation Lines CheckBox-ը (նկար 3.8) նշված է այդ դեպքում բաժանարար գծերը արտապատկերվում են գլխավոր ImageBox-ի վրա, հակառակ դեպքում՝ չեն արտապատկերվում:



նկար 3.8



նկար 3.9

Նկար 3.9-ում արտապատկերված է այն դեպքը, երբ Show Separation Lines CheckBox-ը նշված է:

Ինչպես տեսնում եք նկար 3.9-ում գլխավոր ImageBox-ի վրա արտապատկերված են բաժանարար գծերը:

Բաժանարար գծերը լինում են 2 տեսակի՝

* Օբյեկտների հայտնաբերման համար նախատեսված բաժանարար գծեր և առանձնացված (լոկալ) տարածքներ ներկայացնող գծեր:
* Օբյեկտների հաշվարկի համար նախատեսված բաժանարար գծեր:

Առաջին տեսակի գծերը գծվում են օգտագործելով կապույտ գույնը, իսկ երկրորդ

տեսակի գծերը՝ օգտագործելով կարմիր գույնը:

Շատ հաճախ լինում են դեպքեր երբ 2 տեսակի գծեր համնկնում են:

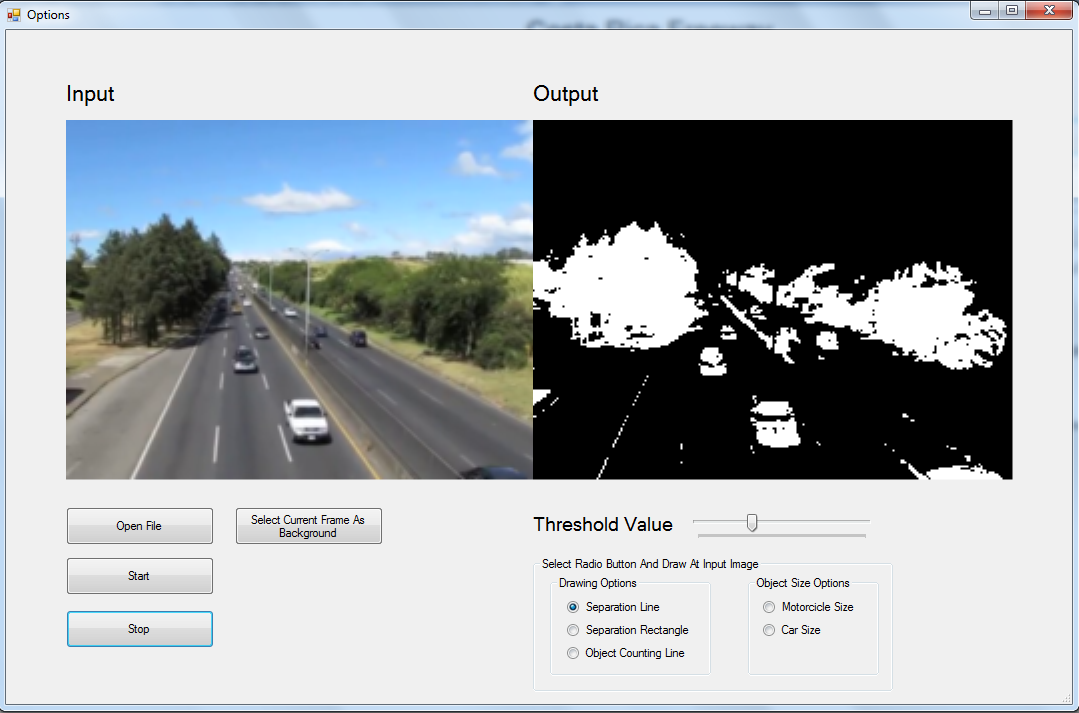
Օրինակ նկար 3.9-ում երևում է, որ գլխավոր ImageBox-ի վրա արտապատկերված կադրում ճանապարհահատվածի ձախ մասում օբյեկտների միայն հաշվարկի գիծ է երևում: Սակայն իրականում այնտեղ կա 2 տեսակի բաժանարար գծերից էլ, ուղակի 2-րդ տեսակի գիծը արտապատկերված է առաջին գծի վրայից, քանի որ նրանք համնկնում են:

Վերը նկարագրված դեպքում օբյեկտի սկզբից հայտնաբերումը այնուհետև նաև քանակի հաշվարկը կատարվել է նույն բաժանարար գիծը անցնելիս:

Իսկ այս նույն ճանապարհահատվածի աջ մասում նշված է կոնկրետ լոկալ առանձնացված տարածք և երբ այդ տարածքը հատում է օբյեկտը՝ այն միանգամից հայտնաբերվում է: Այդ լոկալ տարածքի մի կողմում նշված է նաև օբյեկտների քանակի հաշվարկման բաժանարար գիծ և միայն այդ գծով անցնող օբյեկտներն են հաշվարկվում:

Ինչպես երևում է նկար 3.1-ում և նկար 3.9-ում՝ բոլոր հայտնաբերված օբյեկտները գծանշվում են իրենց եզրաքծերը շրջապատող համապատասխան չափի կանաչ գույնի ուղղանկյունով:

Ինչպես նշվեց վերը Options կոճակը (նկար 3.4) սեղմելուց բացվում է նոր պատուհան կարգավորումների համար: Այդ պատուհանը պատկերված է նկար 3.10-ում:



նկար 3.10

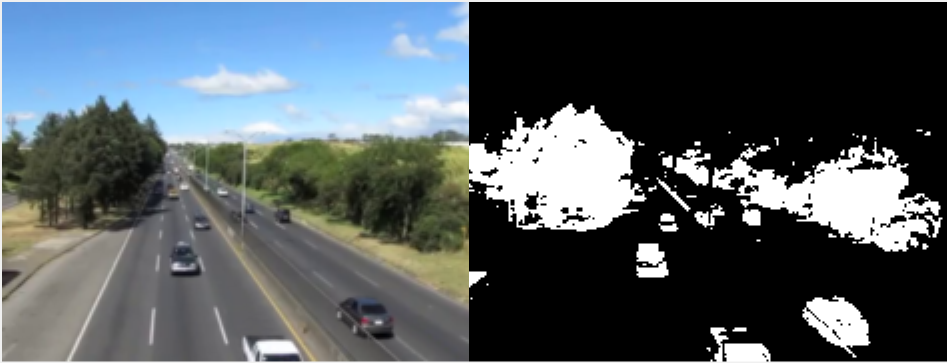
Նկար 3.10-ում պատկերված է Options պատուհանը, որը նախատեսված է ծրագրին անհրաժեշտ կարգավորումներ անելու համար:

Options պատուհանի վերեվի հատվածը ամբողջովին գրավված է 2 ImageBox-երի կողմից, որոնցից ձախինը՝ Input ImageBox-ն է, իսկ աջինը Output ImageBox-ը:

Input ImageBox-ում արտապատկերվում է տեսանյութի ֆայլը իր բնական տեսքով և նաև օգտագործվում է բաժանարար գծերը գծելու, ինչպես նաև օբյեկտների մոտավոր չափերը նշելու համար:

Output ImageBox-ը օգտագործվում է տեսանյութի կադրերի ձևափոխված և մշակված տարբերակը Real-Time-ում ցուցադրելու համար: Տվյալ աշխատանքի դեպքում Output ImageBox-ում արտապատկերվում է հերթական կադրի ֆոնը և օբյեկտները, որոնք համապատասխանաբար ունեն՝ սև և սպիտակ գունանշում:

Input և Output ImageBox-երը առանձին պատկերված են նկար 3.11-ում:

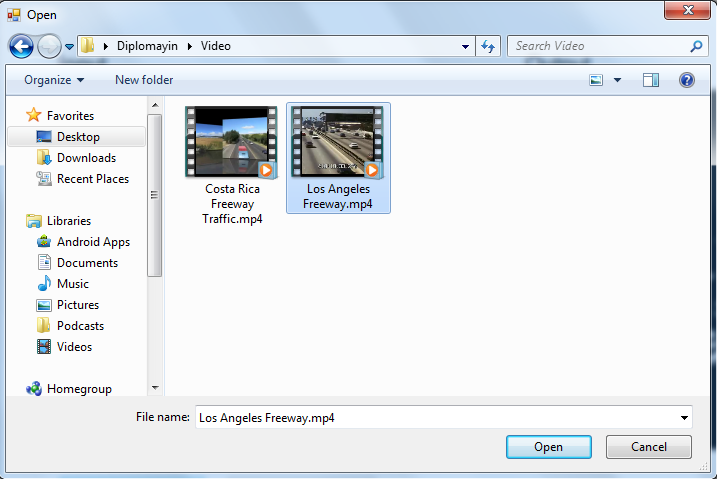


նկար 3.11

Options պատուհանում կա Open File կոճակը (նկար 3.12), որը օգտագործվում է տեսանյութի ֆայլը բացելու համար: Open File կոճակը սեղմելու դեպքում բացվում OpenFileDialog պատուհանը (նկար 3.13) որտեղից օգտագործողը կարող է ընտրել իրեն անհրաժեշտ տեսանյութի ֆայլը:



նկար 3.12



նկար 3.13

Options պատուհանում (նկար 3.10) կան, ինչպես և ծրագրի հիմնական պատուհանում (նկար 3.1) կան Start (նկար 3.5) և Stop (նկար 3.6) կոճակները որոնց նշանակությունները նույնն են ինչ և հիմնական պատուհանում:

Options պատուհանում (նկար 3.10) կա Select Current Frame As Background կոճակ (նկար 3.14) որի օգնությամբ օգտագործողը կարող է ընտրել իրեն հարմար ֆոնային (Background) պատկեր, հետագա ծրագրային գործողությունների համար: Նշենք որ հարմար է նախ և առաջ կանգնեցնել տեսանյութի ցուցադրումը օգտագործելով Stop կոճակը (նկար 3.6), այնուհետև նոր օգտագործել Select Current Frame As Background կոճակ (նկար 3.14):



նկար 3.14

Options պատուհանում (նկար 3.10) կա Threshold Value TrackBar-ը (նկար 3.15), որը օգտագործվում է 2 իրար հաջորդող կադրերի մեջ հանման գործողության արդյունքում ստագված գունային տարբերությունների վորոշ արժեքի որոշման համար, այնպես որ դրանից մեծ գունային արժեք ունեցող կետերը (Pixel) համարվեն օբյեկտին պատկանող կետեր, իսկ փոքր արժեքներ ունեցողները ֆոնին պատկանող կետեր:



նկար 3.15

Threshold Value TrackBar-ի կատարված փոփոխությունները արտապատկերվում են Output ImageBox-ի մեջ: Threshold Value TrackBar-ի տարբեր արժեքների դեպքում ստացված Output մշակված կադրերը կարելի է տեսնել և համեմատել նկար 3.11 և 3.16-ում:



նկար 3.16

Ինչպես երևում է նկար 3.11-ում մեքենաները լիովին իրենց չափով ընդգրկվելեն և ճանաչվել են որպես օբյեկտ, բայց միևնույն ժամանակ ուժեղ քամուց շարժվող ծառը նույնպես ճանաչվել է որպես օբյեկտ: Այս հարցը լուծելու համար կա 2 եղանակ՝

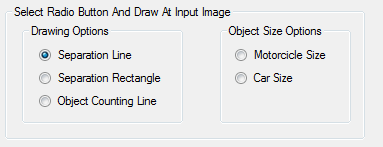
* Threshold Value-ի փոփոխում
* Օբյեկտների հայտնաբերման համար լոկալ տարածքի սահմանում

Առաջին լուծման արդյունքներն ներկայացված են նկար 3.16-ում:

Ինչպես երևում է վերը նշված նկարում ծառը այլևս չի դիտարկվում որպես օբյեկտ,

սակայն միու կողմից մեքենաների հաետնաբերման որակը ընգել է և մեքենան ոչ լիովին է հայտնաբերվել:

Հետևաբար ավելի նպատակահարմար է օգտագործել 2-րդ լուծման ձևը Threshold Value-ն թողնելով անփոփոխ և դրա փոխարեն լոկալ տարածք սահմանելով, այնպես որ մենակ մեքենաները մտնեն այդ տարածքի մեջ, այլ ոչ թե ծառերնել մեկտեղ:



նկար 3.17

Options պատուհանում (նկար 3.10) կան հատուկ RadioButton-ներ վերցված առանձին GroupBox-երի մեջ, որոնց միջոցով օգտագործողը կարող է նշել բաժանարար գծեր, սահմանել լոկալ տարածքներ, գծել քանակի հաշվման բաժանարար գիծը և նաև նշել թիրախային օբյեկտների մոտավոր չափերը:



նկար 3.18

Նկար 3.18-ում պատկերված է բաժանարար գծանշման հատուկ օրինակ, որի արդյունքում կհայտնաբերվեն և կհաշվարկվեն բոլոր այն մեքենաներն ու մոտոցիկլետները, որոնք 3-րդ գծից կանցնեն 2-րդ գիծ կամ հակառակը:

Նկար 3.18-ում պատկերված է բաժանարար գծանշման արդյունքը պատկերված է նկար 3.19-ում:

Հայտնաբերվել է 3-րդ գծից 2-րդ գիծ անցնող մեքենա գծերը փոխելու պահին:



նկար 3.19

**ԳԼՈՒԽ 4 Տեխնիկատնտեսական առաջադրանք**

* 1. **Ծրագրի տեխնիկատնտեսագիտական հիմնավորում**

Տնտեսության բազմաթիվ խնդիրների լուծումը ավելի արագ, հեշտ և էֆեկտիվ է կատարվում, եթե դրանք լուծվում են համակարգչի օգնությամբ` դրա համար մշակված ծրագրի կամ ծրագրային փաթեթի միջոցով: Ծրագիրը կարելի է օգտագործել այնքան անգամ որքան անհրաժեշտ է: Այն պետք է լինի տնտեսապես շահավետ և հարմար օգտագործման տեսանկյունից:

Համակարգիչները լայն կիրառություն ունեն ոչ միայն գիտության մեջ, այլ նաև ժողովրդական տնտեսության տարբեր ճյուղերում:

Էֆեկտիվ արտադրության օպտիմալ կազմակերպման, որակյալ արտադրանքի թողարկման, թողարկվող արտադրանքի ինքնարժեքի իջեցման և այլ ցուցանիշների բարելավման համար անհրաժեշտ է կատարել տնտեսագիտական հաշվարկներ: Դրանից հետո միայն կարելի է կատարել տեխնիկատնտեսական հիմնավորում նախագծվող տարբերակի համար:

Տվյալ բաժնի նպատակն է հանդիսանում որոշել նախագծվող ծրագրի տեխնիկատնտեսական շահավետությունը:

**4.2 Աշխատատարության հաշվարկն առանձին էտապներում**

Հաշվարկները կատարելու համար անհրաժեշտ է հաշվի առնել առանձին էտապների վրա կատարված աշխատանքը, ժամանակը և աշխատողներին :

Աղյուսակում ներկայացվում են աշխատանքի էտապների անվանումներն իրենց համապատասխան տոկոսներով`

Աղյուսակ1

|  |  |  |
| --- | --- | --- |
| **Աշխատանքի էտապների անվանումները** | | **Տոկոսներ (%)** |
| **1** | Օբյեկտների ճանաչման հիմնական գաղափարներ | **5** |
| **2** | Վիդեո ֆորմատների ուսումնասիրություն | **5** |
| **3** | Օբյեկտների առանձնացում ֆոնից | **15** |
| **4** | Օբյեկտների իդենտիֆիկացիա և համարակալում | **10** |
| **5** | Օբյեկտների հետևում և քանակի հաշվարկ | **15** |
| **6** | Երթևեկության կանոնների խախտման գրանցում | **10** |
| **7** | Ծրագրային ինտերֆեյսի կազմակերպում | **5** |
| **8** | Ճյուղային էկանոմիկա | **15** |
| **9** | Կենսագործնեության անվտանգություն | **10** |
| **10** | Բնապահպանություն | **5** |
| **11** | Գրականություն | **5** |

Հնգօրյա աշխատանքային ռեժիմի դեպքում ունենք շաբաթական 40ժ, ամսեկան` 176ժ, ամբողջ աշխատանքը 7 ամսում կլինի 1232ժ :

Հաշվենք յուրաքանչյուր էտապի աշխատատարությունը`

|  |  |  |  |
| --- | --- | --- | --- |
| T1 = | 1232 \* 5 / 100 = | 61.6 | (մարդ/ժամ) |
| T2 = | 1232 \* 5 / 100 = | 61.6 | (մարդ/ժամ) |
| T3 = | 1232 \* 15 / 100 = | 184.8 | (մարդ/ժամ) |
| T4 = | 1232 \* 10 / 100 = | 123.2 | (մարդ/ժամ) |
| T5 = | 1232 \* 15 / 100 = | 184.8 | (մարդ/ժամ) |
| T6 = | 1232 \* 10 / 100 = | 123.2 | (մարդ/ժամ) |
| T7 = | 1232 \* 5 / 100 = | 61.6 | (մարդ/ժամ) |
| T8 = | 1232 \* 15 / 100 = | 184.8 | (մարդ/ժամ) |
| T9 = | 1232 \* 10 / 100 = | 123.2 | (մարդ/ժամ) |
| T10 = | 1232 \* 5 / 100 = | 61.6 | (մարդ/ժամ) |
| T11 = | 1232 \* 5 / 100 = | 61.6 | (մարդ/ժամ) |

Ունենալով աշխատատարությունը` կարող ենք հաշվել աշխատանքի ցիկլի տևողությունը օրերով, հետևյալ բանաձևի մջջոցով`

ti=Ti \*K/Fիր\*Ri (1),

որտեղ`

Ti - i-րդ էտապի աշխատատարությունն է, մարդ/ժամ,

Ri -տվյալ էտապում աշխատողների թիվն է,

K -օրացուցային օրերի` բանվորական օրերի փոխարինման գործակից(K=1.38),

Fիր.-մեկ աշխատողի իրական ժամանակային ֆոնդ,ժամ/օր, Fիր =Fն (1-α/100)),

Fն.-ժամանակի նոմինալ ֆոնդ, Fն. = 8ժամ/օր,

α-աշխատանքային ժամանակի պլանավորված կորուստներ, α=5 - 10%,

α-ն ընդունենք 7% և կատարենք հաշվարկը`

Fիր .=8(1-7/100) =7.44(ժամ/օր),

Fիր.-ի արժեքը տեղադրենք (1) - ի մեջ, Ri -ն ընդունելով 1`

|  |  |  |
| --- | --- | --- |
| t1 = | 61.6 \* 1.38 / 7.44 = | 11 |
| t2 = | 61.6 \* 1.38 / 7.44 = | 11 |
| t3 = | 184.8 \* 1.38 / 7.44 = | 34 |
| t4 = | 123.2 \* 1.38 / 7.44 = | 23 |
| t5 = | 184.8 \* 1.38 / 7.44 = | 34 |
| t6 = | 123.2 \* 1.38 / 7.44 = | 23 |
| t7 = | 61.6 \* 1.38 / 7.44 = | 11 |
| t8 = | 184.8 \* 1.38 / 7.44 = | 34 |
| t9 = | 123.2 \* 1.38 / 7.44 = | 23 |
| t10 = | 61.6 \* 1.38 / 7.44 = | 11 |
| t11 = | 61.6 \* 1.38 / 7.44 = | 11 |

**4.3 Աշխատավարձերի հաշվարկ**

Յենթադրում ենք, որ ամբողջ աշխատանքի համար մեզ անհրաժեշտ են հետևյալ աշխատողները`

1. ծրագրավորող,
2. ինժեներ,
3. ինժեներ-տնտեսագետ,
4. ինժեներ-բնապահպան,
5. կենսանվտանգության ինժեներ

Հետևյալ մասնագետների միջին աշխատավարձը ընդունում ենք հետևյալ կերպ`

|  |  |  |
| --- | --- | --- |
|  | ***մասնագիտություն*** | ***ՀՀ դրամ*** |
| 1. | ծրագրավորող | 200.000 |
| 2. | ինժեներ | 70.000 |
| 3. | ինժեներ-տնտեսագետ | 80.000 |
| 4. | ինժեներ-բնապահպան | 80.000 |
| 5. | կենսանվտանգության ինժեներ | 80.000 |

Յուրաքանչյուր էտապի աշխատատարությունը և ստացված տվյալները հաշվի առնելով կազմենք աշխատավարձերի որոշման և աշխատողների թվի որոշման աղյուսակը`

Աղյուսակ 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ախատանքի էտապների անվանումները | Աշխատողի անվանում | Ti | Չափը (դրամ)  1 մարդու  համար  Ընդհանուր | | Աշխատողների թիվը |
| Օբյեկտների ճանաչման հիմնական գաղափարներ | ինժեներ | 61.6 | 389 | 23960 | 1 |
| Վիդեո ֆորմատների ուսումնասիրություն | ինժեներ | 61.6 | 389 | 23960 | 1 |
| Օբյեկտների առանձնացում ֆոնից | ծրագրավորող | 184.8 | 1111 | 205313 | 2 |
| Օբյեկտների իդենտիֆիկացիա և համարակալում | ծրագրավորող | 123.2 | 1111 | 136875 | 1 |
| Օբյեկտների հետևում և քանակի հաշվարկ | ծրագրավորող | 184.8 | 1111 | 205313 | 2 |
| Երթևեկության կանոնների խախտման գրանցում | ծրագրավորող | 123.2 | 1111 | 136875 | 1 |
| ծրագրային ինտերֆեյսի կազմակերպում | ծրագրավորող | 61.6 | 1111 | 68437 | 1 |
| Ճյուղային էկանոմիկա | ինժեներ-տնտեսագետ | 184.8 | 444 | 81873 | 1 |
| Կենսագործնեության անվտանգություն | կենսանվտանգության ինժեներ | 123.2 | 444 | 54700 | 1 |
| Բնապահպանություն | ինժեներ-բնապահպան | 61.6 | 444 | 27350 | 1 |
| Գրականություն | ինժեներ | 61.6 | 389 | 23960 | 1 |
| Ընդհանուր | - | - | - | 988615 | 13 |

Տվյալ աշխատանքի համար անհրաժեշտ աշխատողների թիվը ստանում ենք 13, որը սակայն ֆինանսական տեսանկյունից ձեռնտու չէ, հետևաբար աշխատողների թիվը կրճատում ենք` մյուս աշխատողներին լրացուցիչ աշխատանք հանձնարարելով: Որպես օպտիմալ տարբերակ ընդունում ենք բարձր որակավորում ունեցող ծրագրավորողին և ինժեներին` հանձնարարելով մյուս աշխատողների աշխատանքը ևս:Այդ դեպքում նրանց աշխատավարձերը կլինեն`

**Aինժ. =** 23960 + 23960 + 81873 + 54700 + 27350 + 23960 = 235803 (ՀՀ դրամ)

**Aծրագ. =** 205313+ 205313+ 136875 + 136875 + 68437 = 615938 (ՀՀ դրամ)

1 ամսում ինժեներին պետք է վճարել` 235803 / 7 = 33686 (ՀՀ դրամ):

Ծրագրավորողին` 615938 / 7 = 87991 (ՀՀ դրամ):

**4.4 Աշխատանքի համար անհրաժեշտ տեղամասի վարձակալման հաշվարկ**

Հաշվի առնելով աշխատողների քանակը, անհրաժեշտ գույքի չափսերը` նպատակահարմար է վարձակալել 20մ2 մակերես ունեցող տարածք:

**Sկենց. = Sաշխ.\*6%=1,2(մ2)**

**Sընդ. = Sաշխ.+ Sկենց. =20+1,2=21,2(մ2)**

Կարածքի վարձակալման հաշվարկի բանաձևը հետևյալն է`

**Pմակ.1=T\*P\*Pմակ.,**

որտեղ`

**T**-վարձակալման ամիսների թիվը(7),

**P**-1(մ2) տարածքի վարձակալման ամսեկան գումարը(3500 ՀՀ դրամ)

**Pմակ.- Sընդ**.-ն է (մ2)

**Pմակ.1=7\*21.2\*3500=519400 (ՀՀ դրամ):**

**4.5 Խմելու ջրի ծախսի հաշվարկ**

Ընդունված է, որ մջինում մարդն օրեկան ծախսում է` V=30(լ) ջուր:ՀՀ-ում խմելու ջրի սակագինը 1(մ3)-ի դիմաց կազմում է 181(ՀՀ դրամ),որը կնշանակենք P-ով:

Ջրի ծախսի ընդհանուր բանաձևը հետևյալն է`

**Pջուր. =n\*T\*T1\*V\*p (3),**

որտեղ՝

**n-**աշխատողների քանակը

**T-**աշխատանքային ամիսների թիվն է

**T1-**աշխատանքային օրերի թիվը**(**1 ամսվա մեջ**):**

Մեր տվյալները տեղադրելով` կստանանք`

**Pջուր. =2 \* 7 \* 22.5\* 0.03 \* 181=1710 (ՀՀ դրամ)**

**4.6 Էլեկտրաէներգիայի ծախսը**

Այս հաշվարկը կատարվում է ենթակետերով:

**4.6.1 Տարածքի ջեռուցման էլէկտրաէնէրգիայի ծախսը**

Տարածքի հովացման և տաքացման համար տեղադրվում է Mitsubishi օդափոխիչ: Ընդհանուր բանաձևն է`

**P օդ. = T\*Kօդ\* P\*K (4),** որտեղ

**T-** օդափոխիչի աշխատած ժամերի քանակը,

**K օդ.-** 1ժ-ում օդափոխիչի ծախսած էներգիայի քանակը**(կՎտ\*ժ)**

**P-**1կՎտ էներգիայի արժեքը ՀՀ-ում

**K-** օդափոխիչի չաշխատած ժամերը հաշվի առնող գործակից

Մեր պարագայում **T=1232(ժամ), Kօդ. =3,2(կՎտ\*ժ), P=30(**ՀՀ դրամ**),** իսկ **K=0,6:**

**Pօդ. =1232 \* 3,2 \* 30 \* 0,6=70970(ՀՀ դրամ)**

**4.6.2 Համակարգչի էլէկտրաէնէրգիայի ծախսը**

Ընդհանուր բանաձևը հետևյալն է`

**Pհամ = T\*Kհամ\*p\*n (5),**

որտեղ՝

**T-** համակարգչի աշխատած ժամերի քանակն է

**Kհամ-** 1ժ-ում համակրգչի ծախսած էներգիայի քանակը**(**կՎտ\*ժ**)**

**p –** ՀՀ-ում 1կՎտ էներգիայի արժեքը

**n –**համակարգիչների թիվը

Մեր դեպքում **T=1232 (**ժամ**), Kհամ=0,15(**կՎտ\*ժ**), p=30(**ՀՀ դրամ**),n=2**

Տվյալները տեղադրելով **(**5**)** բանաձևի մեջ` կստանանք

**Pհամ. =1232 \* 0,15 \* 30 \* 2=11090 (ՀՀ դրամ)**

**4.6.3 Տպիչի էլէկտրաէնէրգիայի ծախսը**

Հաշվակը կատարելիս պետք է հաշվի առնել, որ տպիչը աշխատելու է միջին հաշվով օրեկան 0,05 ժամ:Հաշվարկը կատարվում է հետևյալ բանաձևով`

**Pտպ. = T\*Kտպ\*p\*n (6),** որտեղ

**T –** տպիչի աշխատած ժամերի քանակը

**Kտպ–** 1ժ-ում տպիչի ծախսած էներգիայի քանակը(կՎտ\*ժ)

**p** – ՀՀ-ում 1կՎտ էներգիայի արժեքը

**n –** տպիչների թիվը

Մեր դեպքում **T= 10(ժամ), Kտպ.=0,6(կՎտ\*ժ),p=30(ՀՀ դրամ), n=1:** Ունեցած տվյալները տեղադրելով (6) բանաձևի մեջ` կստանանք

**Pտպ. = 0,6 \* 30 \* 10 \* 1 =180(ՀՀ դրամ)**

**4.6.4 Տարածքի լուսավորության էներգիայի ծախսը**

Ելնելով տարածքի չափսերից և կատարվող աշխատանքի բնույթից` նպատակահարմար է լուսավորության համար տեղադրել 5 լամպ, յուրաքանչյուրը 60Վտ հզորության:Հաշվի է առնվում նաև այնփաստը, որ աշխատանքային ժամերի 40%-ի լուսավորվածությունը չի բավարորում աշխատանքի պայմանններին: Հաշվարկը կկատարենք հետևյալ բանաձևի մջջոցով`

**Pլուս = T\*Kլուս\*p\*n (7),**

որտեղ՝

**T–** լուսավորվող ժամերի քանակը

**Kլուս –** 1ժ-ում լամպի ծախսած թներգիայի քանակը(կՎտ\*ժ)

**p–** ՀՀ-ում 1կՎտ էներգիայի արժեքը

**n–** լամպերի քանակը

**T= 504**ժ**, Kլուս.=0,06(**կՎտ\*ժ**), p=30 (ՀՀ դրամ), n=5**:Այս տվյալները տեղադրելով (7) բանաձևի մեջ` կստանանք

**Pլուս =504 \* 0,06 \* 30 \* 5=4536(ՀՀ դրամ)**

Էլեկտրաէներգիայի ամբողջ ծախսը կլինի`

**Pէլ. = Pօդ.+Pհամ.+Pտպ.+Pլուս. (8)**

**Pէլ. = 75580+11340+180+4536 = 91640 (ՀՀ դրամ):**

**4.7 Հեռախոսային վճարումների հաշվարկ**

ՀՀ-ում հեռախոսային խոսակցությունների բաժանորդային վարձը կազմում է 1100 ՀՀ դրամ 1 ամսվա համար:Լրացուցիչ հեռախոսային խոսակցություններին կհատկացնենք 5000ՀՀ դրամ:Ընդհանուր ծախսը կհաշվարկենք այս բանաևով`

**Pհեռ = T\* ( Pբաժ.+Pլր.) (9),**

որտեղ՝

**T–ամիսների թիվը**

**Pբաժ. =1100(ՀՀ դրամ)**

**Pլր = 5000(ՀՀ դրամ)**

**Pհեռ = 7\*( 1100+5000) =42700(ՀՀ դրամ)**

**4.8 Կոմունալ վճարումների ընդհանուր հաշվարկ**

Ընդհանուր հաշվարկը կկատարվի այսպես`

**Pկոմ= Pմակ.1 + Pջուր. + Pէլ. + Pհեռ=519400 +1710 + 91640 + 42700 = 655450 (ՀՀ դրամ)**

**4.9 Արտադրական գույքի ծախսերի հաշվարկ**

Ներկայացնենք աղյուսակի տեսքով`

Աղյուսակ 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Գույքի անվանում | Քանակ (հատ) | 1 հատի արժեքը առանց ԱԱՀ (ՀՀ դրամ) | Ընդհանուր առանց ԱԱՀ (ՀՀ դրամ) | Ընդամենը առանց ԱԱՀ (ՀՀ դրամ)  **725.000** |
| Intel Core i5, 8GB RAM, 1TB HDD, DVD-RW | 2 | 260000 | 520000 |
| Monitor LG LED 22'' | 2 | 75000 | 150000 |
| Keyboard genius kb200 | 2 | 2500 | 5000 |
| Mouse genius xserol | 2 | 2000 | 4000 |
| Printer Canon LBP2900B | 1 | 46000 | 46000 |

**4.10 Տնտեսական գույքի ծախսերի հաշվարկ**

Ներկայացնենք աղյուսակի տեսքով՝

Աղյուսակ 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Գույքի անվանում | Քանակ  (հատ) | 1 հատի արժեքը առանց ԱԱՀ (ՀՀ դրամ) | Ընդհանուր առանց ԱԱՀ (ՀՀ դրամ) | Ընդամենը առանց ԱԱՀ (ՀՀ դրամ)  **260.000** |
| Օդապոխիչ Mitsubishi | 1 | 166000 | 166000 |
| Հեռախոս Panasonic | 1 | 9000 | 9000 |
| Սեղան համակարգչային | 2 | 15000 | 30000 |
| Գրասեղան | 1 | 20000 | 20000 |
| Աթոռ | 5 | 7000 | 35000 |

**4.11 Ամբողջ աշխատանքի վրա կատարված ծախսերի հաշվարկ**

Նախորդ կետերում արդեն կատարվել են բոլոր տիպի ծախսերի հաշվարկները, այժմ պետք է ներկայացնենք դրանց գումարն ու հաշվի առնենք նաև շահութահարկը:

Աղյուսակ 5

|  |  |  |  |
| --- | --- | --- | --- |
| Կատարված ծախսի անվանում | Արժեքը  առանց ԱԱՀ  (ՀՀ դրամ) | Ընդհանուր  առանց ԱԱՀ  (ՀՀ դրամ)  2.497.191 | Ընդհանուր + 20% շահութահարկ  (ՀՀ դրամ)  2.996.629 |
| Աշխատավարձ | 851.741 |
| Pկոմ | 655.450 |
| Արտադրական գույք | 725.000 |
| Տնտեսական գույք | 260.000 |
| Գրենական պիտույքներ | 5.000 |

**4.12 Եզրակացություն**

Հաշվարկները ցույց տվեցին, որ կատարված աշխատանքը կարող ենք վաճառել 2.996.629 ՀՀ դրամից ոչ պակաս գումարով, հակառակ դեպքում շահույթ չենք ունենա:

**ԳԼՈՒԽ 5 Կենսագործունեության անվտանգություն**

* 1. **Կենսագործունեության միջավայրում նախագծվող օբյեկտներից առաջացող հնարավոր վտանգավոր և վնասակար գործոնների վերլուծությունը և դրանց վերացման կամ նվազեցման միջոցառումները**

ԷՀՄ-ներով համալրված սենյակներում վտանգավոր գործոններ կարող են հանդիսանալ հետևյալ գործոններըª

1. Աղմուկի բարձր մակարդակը
2. Էլեկտրական հոսանքը
3. Հրդեհները
4. Տարածքի արհեստական լուսավորվածության անբավարարվածությաւոնը
5. Բնական լուսավորվածության անբավարարվածությունը կամ բացակայությունը:
6. Ճառագայթումը

1. Աղմուկը աշխատանքային սենյակում լայն տարածված վտանգավոր գործոններից է, որի ազդեցությունը չի սահմանափակվում միայն լսողական օրգանների վրա, նյարդային համակարգի միջոցով, այն ազդում է նաև ներքին օրգանների վրա: Բարձր աղմուկի պայմաններում աշխատող մարդիկ բողոքում են գլխացավերից, գերհոգնածությունից, անքնությունից, որն էլ դառնում է աշխատանքի արտադրողականության անկման պատճառ:

Աղմուկի դեմ պայքարի արդյունավետ միջոցներից է ձայնակլանիչ հատկություններով օժտված նյութերի օգտագործումը: Փակ տարածության մեջ ձայնի ալիքները հասնում են պատերին, առաստաղին կամ այլ խոչընդոտներին, որոնք ոչ միայն կլանում այլև անդրադարձնում են ձայնային էներգիան: Աղմուկից պաշտպանվելու համար գոյություն ունեն անհատական միջոցներ հակաաղմուկային ականջակալներ, խցաններ և այլն:

Աղմուկի աղբյուրներ կարող են հանդիսանալ տվյալ և հարակից տարածքներում աշխատող էլեկտրական, էլեկտրոնային և մեխանիկական սարքեր և սարքավորումներ: Օրինակ օդափոխիչները, տպիչները և այլն:

2. Էլեկտրական սարքավորումները, որոնց թվին են պատկանում ԷՀՄ-ի բոլոր սարքերը մարդու համար մեծ վտանգ են ներկայացնում, քանի որ շահագործման պրոցեսում մարդը կարող է դիպչել այնպիսի մասերին, որոնք գտնվում են լարման տակ:

Էլեկտրական հոսանքի վտանգավորության պատճառներ են հանդիսանումª վնասված հոսանքակիր հաղորդալարերը, ԷՀՄ-ի իրանը և այլ սարքավորումներ, որոնք գտնվում են լարման տակ: Վատ հողանցման, հատակի ոչ ճիշտ ընտրության դեպքում մարդը կարող է ընկնել քայլային լարման տակ, որը հետևանք է վնասված հաղորդալարերի:

Էլեկտրական վնասվածքների կանխարգելման համար կարևոր նշանակություն ունի էլեկտրաանվտանգության ապահովումը:

Հոսանքահարումից պաշտպանվելու համար օգտագործվում են մեկուսիչ յուղեր: Հոսանքահար մասերը փակվում են զանազան մեկուսիչ նյութերով: Սարքերն ու սարքավորումները հողանցում են: Հոսանքահարվածին անհրաժեշտ է ցուցաբերել առաջին օգնություն, այնուհետև դիմել բժշկի:

3. Հրդեհի ծագման ու տարածման ժամանակ մեծ դեր են խաղում արտադրական շենքերի նախագծական առանձնահատկություններըª չափերը, օգտագործված նյութերի տեսակը և այլն:

Ձեռնարկության նախագծման ընթացքում հաշվի է առնվում մարդկանց անվնաս էվակուացման հնարավորությունները: Հնարավորինս կարճ ժամկետում մարդիկ պետք է ազատեն շենքը` անցնելով աշխատատեղից մինչև անվտանգ գոտի տանող ճանապարհը: Այն չպետք է անցնի թեք ճանապարհով, պարուրաձև աստիճաններով, խոչընդոտներով:

Հրդեհային անվտանգության կարևոր պայմաններից է հրդեհի առաջին իսկ նշանների դեպքում անմիջապես ահազանգումը հրշեջ ծառայություն: Հրդեհը վերացնելեւ գործում մեծ դեր ունի հիմնարկության կամավոր հակահրդեհային պաշտպանությունը:

ԷՀՄ-ներով համալրված սենյակներում հրդեհներ կարող են առաջանալª էլեկտրական հաղորդալարերից, էլեկտրական սարքերի խափանումներից, մարդկանց անփույթ վարվելակերպից, ջեուցման սարքերից (գազ):

4. Արհեստական և բնական լուսավորվածության անբավարարվածությունը մարդկանց մոտ առաջացնում է տեսողական օրգանների գերլարվածություն և հոգնածություն:

Անբավարար լուսավորվածության պատճառ կարող են հանդիսանալ լուսավորման միջոցների ոչ ճիշտ հաշվարկը և տեղադրումը:

Որպես արհեստական լույսի աղբյուր հանդիսանում են շիկացման թելիկով և գազապարպման լամպերը: Լամպերը կոտրվելուց ինչպես նաև լամպի լույսը տվյալ աշխատանքային տեղը ուղղելու և աշխատողին լամպի շլացուցիչ լույսից պաշտպանելու համար օգտագործում են արմատուրա և այդ լամպ արմատուրա միացությունը անվանում են լուսամփոփ:

Բնական լուսավորվածության բացակայությունը կամ անբավարվածությունը աշխատողների մոտ առաջացնում է գերլարվածություն, տեսողական օրգանների գերհոգնածություն:

Բնական լուսավորվածության բացակայությունը կամ անբավարվածությունը առաջանում են պատուհանների բացակայության կամ նրանց չափերի և դիրքի ոչ ճիշտ ընտրության պատճառով:

5. Ճառագայթումից պաշտպանվելու համար օգտագործում են մանիտորների համար նախատեսված հատուկ պաշտպանիչ ֆիլտրեր, որոնք իրենցից ներկայացնում են ապակե էկրաններ որոնք հողանցված են: Ցանկալի է օգտագործել 15” և 17” մանիտորներ քանի որ նրանց էկրաններն իրենց մեջ պարունակում են վերը նշված ֆիլտրերը:

Ճառագայթումը մարդկանց մոտ առաջացնում է տեսողական օրգանների լարվածություն և հոգնածություն: Երկարատև ճառագայթումը կարող է առաջացնել տեսողության թուլացում նույնիսկ նաև կուրացում:

**5.2 Բնական լուսավորության կազմակերպումը համակարգչային ստահում**

**5.2.1 Արտադրական Լուսավորություն**

Արտադրական լուսավորության անթերի կազմակերպումն աշխատանքային բարենպաստ պայմանների ստեղծման կարևոր գործոն է:

Թույլ լուսավորությունը ստիպում է մարդուն լարեու տեսողությունը, առաջ է բերում ընդհանուր, և մասնավորապես տեսողական օրգանների հոգնածություն:

Որքան թույլ է լուսավորվածությունը, այնքան ավելի փոքր է աչքի տեսողական սրությունը, այսինքն` առավել մանր առարկաները տարբերելու ունակությունը: Թույլ լուսավորվածությունը նվազեցնում է նաև տեսողությամբ ընկալելու արագությունը և աչքի կոնտրաստային զգացողությունը:

Աչքն ընդունակ է հարմարվելու տվյալ լուսավորվածությանը և պայծառությանը:

Հարմարվելու այդ գործողությունը, որը աչքի ադապտացիա է կոչվում, կարող է տևել նույնիսկ մինչև 50 րոպե: Այս ժամանակամիջոցում աչքի տեսողական ունակությունը խիստ նվազում է: Աչքի հաճախակի ադապտացիան ոչ միայն վնասակար է և հոգնեցուցիչ, այլ լարված աշխատանքում ստեղծում է վտանգավոր պահեր և վթարների պատճառ դառնում:

Անհավասարաչափ կամ թույլ լույսի տևական ազդեցությունը կարող է նաև տեսողության մասնիկի կորստի և մի շարք այլ հիվանդությունների պատճառ դառնալ:

Անբարենպաստ լուսավորությունը զգալի չափով նվազեցնում է աշխատանքի արտադրողականությունը, մեծացնում է խոտանի տոկոսը և իջեցնում արտադրանքի որակը:

Լուսավորությունը ունի նաև գեղագիտական նշանակություն: Լավ կազմակերպված լուսավորության պայմաններում աշխատանքը ավելի հաճելի և դյուրին է թվում:

Լուսավորությունը պետք է լինի բավարար և իր սպեկտրով նմանվի բնական լույսին:

Արտադրական շենքում և աշխատատեղերում թանձր ստվերներ կամ կուրացնող պայծառության փայլեր չպետք է լինեն:

Լույսի աղբյուրը, ինչպես և դրանց համար անհրաժեշտ սարքավորումները, պետք է հաճելի տեսք ունենան, լինեն անվնաս և ընտրվեն տնտեսապես նպաստավոր միջոցներով:

**5.2.2 Բնական Լուսավորության Առանձնահատկությունները**

Բնական լույսը մարդու համար կենսական անհրաժեշտություն է, այն ցրվում է հավասարաչափ և տեսողության համար բարենպաստ է: Հազարամյակների ընդացքում նրան սովորել ու հարմարվել է մարդու աչքը:

Բնական լույսը հանգստացնում է մարդու նյարդային համակարգը, բարձրացնում տրամադրությունն ու աշխատունակությունը, ստեղծում բնության հետ անմիջական կապի զգացողություն:

Այդ պատճառով բնական լուսավորության ճիշտ կազմակերպումը տեխնիկատնտե-

սական նշանակության հետ միասին ունի նաև հիգիենիկ մեծ նշանակություն:

Բնական լույսով լուսավորվածության աստիճանը խիստ փոփոխական է: Տարվա եղանակի, օրվա ժամերի, ամպամածության, օդի մաքրության և այլ պայմաններից կախված այն փոխվում է:

Արտադրական շենքերի բնական լուսավորվածությունը տրվում է հետևյալ ձևերով`

Կողքից` պատուհաններից կամ թափանցիկ նյութերից պատրաստված պատերից:

Վերևից` առաստաղում լուսավորության նպատակով պատրաստված ապակեծածկ մասերից (երդիկից):

Միաժամանակ պատուհաններից և երդիկից (կոմբինացված):

Այս կամ այն ձևի ընտրությունը կախված է շենքի չափերից, արտադրության բնույթից և մի շարք այլ գործոններից:

Քանի որ բնական լույսի ինտենսիվությունը խիստ փոփոխական է, ուստի բնական լույսով շենքերի լուսավորվածության ատիճանը գնահատվում է հարաբերական մեծությամբ` բնական լուսավորվածության գործակցով (ԲԼԳ):

Շենքի ներսում, որևէ կետի բնական լուսավորվածության հարաբերությունն է այդ նույն ժամանակ դրսում ամբողջ երկնակամարից ցրված լույսով հորիզոնական հարթության լուսավորությանը` արտահայտված տոկոսներով.

որտեղ` –ը ներսում, M կետի լուսավորվածության գործակիցն է:

-ն տվյալ M կետի լուսավորվածությունն է շենքի ներսում,

-ն դրսում հորիզոնական հարթության լուսավորվածությունն է բաց երկնակամարից:

Բնական լուսավորվածության գործակցի ամենափոքր հաշվարկային արժեքը որոշվում է դրսում 5000լք լուսավորվածության պայմաններում:

**5.2.3 Բնական Լուսավորության Նորմերը**

Բնական լուսավորության գործակցի մեծությունը որոշելիս պետք է հաշվի առնել նաև արտադրական շենքի աշխարհագրական տեղը և դիրքը`

որտեղ` –ը ԲԼԳ - ի հաշվարկային մեծությունն է,

–ն ԲԼԳ - ի նորմավորված արժեքն է, որն ընտրվում է վերևի աղյուսակից:

m–ը լուսային կլիմայի գործակիցն է, որի մեծությունը ընտրվում է տեղեկագրքերից և կախված է նրանից, թե որտեղ է գտնվում արտադրական շեմքը:

c –ն կոչվում է արևայնության գործակից: Այն կախված է արեգակի նկատմամբ շենքի գրաված դիրքից և տեղադրման վայրից:

**5.2.4 Բնական Լուսավորության Հաշվարկը Աշխատանքային Տեղերում**

Աշխատասենյակում, որի երկարությունը 8մ է, լայնությունը` 5մ և բարձրությունը` 3,2մ հաշվել բնական լուսավորությունը սանիտարական պահանջվող նորմերով ապահովելու համար անհրաժեշտ լուսամուտների ընդհանուր մակերեսը և քանակը:

Նախ պետք է որոշել կատարվող տեսողական աշխատանքի կարգը և բնական լուսավորության գործակիցը այն գոտու համար, որտեղ գտնվում է տվյալ հիմնարկը: Տեսողական աշխատանքը երրորդ կարգի է` բարձր ճշտության:

Հայաստանը գտնվում է ընդունված հինգ գոտիներից հինգերորդում, իսկ բնական լուսավորվածության նորման ըստ նորմերի (CH) տրված է երրորդ գոտու համար,այսինքն` միջին գոտու հաշվով:

որտեղ` , m–լուսային կլիմայի գործակիցն է առանց հաշվի առնելու արևային լույսի ուղիղ հոսքը (l-1,2;ll-1,1;lll-1,0;lV-0,9;V-0,8 – ըստ սանիտարական նորմերի),

m=0.8, C- արևայնության գործակից, հաշվի առնելով արևի լույսի ուղիղ հոսքը` C=1:

Հետևաբար`

Բնական լուսավորության հաշվարկը աշխատանքային տեղում կատարվում է սենյակի լուսամուտների ընդհանուր մակերեսի որոշմամբ.

որտեղ`

k- ն պահեստային գործակից (ապակիների կեղտոտվածության և այլն),

ղ - ն լուսամուտների լուսային բնութագիրը, որը որոշվում է ըստ սենյակի պարամետրերի և լուսամուտի վերին մասից մինչև աշխատանքային մակերեսը ունեցած բարձրության հարաբերությամբ՝ a/b և b/hաշխ. A - երկարություն` 8մ, b - լայնություն` 5մ□(⇒┴ ) a/b= 8/5=1,6;

- հաշվարկային բարձրությունն է, որը որտեղ` hլ-ն առաստաղից մինչև լուսամուտի վերին մասը ունեցած հեռավորությունն է: Մեր դեպքում hլ=0,3մ; hմ- ն հատակից մինչև աշխատանքային մակերեսը եղած բարձրությունն է, hմ=0,8մ:

Այսպիսով՝

Որքան a/b հարաբերությունը փոքր է b/h-ից, այնքան -ն մեծ է(ըստ աղյուսակի մինչև =66): 1,6/2,4=0,7 այս դեպքի համար =46:

լուսաթափանցման ընդհանուր մակերեսի գործակից, ըստ ապակու որակի, շրջանակի տիպի, արևապաշտպան սարքերի և այլն:

որտեղ`

-ն ապակու լուսաթափանցման գործակիցն է, սովարական միաշերտ ապակիների դեպքում =0.9,

- գորշակից, որը հաշվի է առնում լույսի կորուստները ըստ շրջանակի տիպի. փայտից լինելու դեպքում` 0,75 է, մետաղականը` 0,6,

- լույսի կորուստները հաշվի առնող գործակից. լինում է 0,5փ 0,8 (ըստ կեղտոտվածության),

- շինության տանող կոնստրուկցիաներից առաջացող լույսի կորուստները հաշվի առնող գործակից (սյուներ, կողային պատեր և այլն), =1,

- գործակից. արևապաշտպան սարքերից լույսի կորուստները հաշվի առնող =1:

- գործակից է ըստ a,b,H հարաբերության և դրանց մակերեսի անդրադարձման չափերի. սպիտակեցված առաստաղի դեպքում =0.7; սպիտակեցված պատերի դեպքում =0.5; փայտե հատակ` չներկված =0.9 և a/b և b/hաշխ հարաբերությունը` 0,7-ի: Այս արժեքների դեպքում ըստ սանիտարական նորմերի =6 (լինում է 1փ10):

- գործակից, որը հաշվի է առնում դիմացի շենքի հեռավորությունից գցած ստվերը: Եթե շենքը շատ մոտ չէ =1,7; բացակայության դեպքում =1:

Այսպիսով`

Լուսամուտների քանակը որոշվում է  հարաբերությանը համապատասխան` ըստ տեսողական աշխատանքի կարգի: Տեսողական աշխատանքը մեր դեպքում բարձր բարձր ճշտության է և այդ հարաբերությունը պետք է բավարարի -ին:



Այս հարաբերությունը բավարար է ապահովելու համար անհրաժեշտ լույսի քանակը բարձր ճշտության աշխատանք կատարելու համար:

**ԳԼՈՒԽ 6 Բնապահպանություն**

**6.1 Մթնոլորտի աղտոտումը գազային արդյունաբերություններում**

Բնական պայմաններում օդի քիմիական բաղադրությունը շատ չնչին է փոփոխվում (ազոտ - 78%, թվածին - 21%, ածխածնի երկօքսիդ, արգոն, այլ իներտ գազեր, ազոտի երկօքսիդ, ծծումբի օքսիդներ, ջրային գոլորշի, ամոնիակ, ջրածին, օզոն, մեթան - գումարային մոտավորապես 1%): Սակայն մարդու տնտեսական և արտադրական գործունեության հետևանքով տեղի է ունենում մթնոլորտի բաղադրության զգալի փոփոխություն, հատկապես փոխվում է երկրորդային գազային խառնուրդների (ծծմբի դիօքսիդ՝ SO2, ազոտի օքսիդներ՝ NO և NO2, ածխածնի մոնօքսիդ՝ CO, մեթան՝ CH4 և այլն) խտությունը: Մեծանում է նաև պինդ կախված մասնիկների խտությունը, այսինքն տեղի է ունենում մթնոլորտային օդի աղտոտում: Այն ավելի արտահայտվում է մեծ քաղաքներում և արդյունաբերական կենտրոներում:

Իսկ որո՞նք են մթնոլորտային օդի աղտոտման հիմնական մարդածին աղբյուրները: Դրանք, առաջին հերթին, էներգիայի արտադրությունն է, ավտոտրանսպորտը, սև և գոնավոր մետալուրգիան, քիմիական և նավթաքիմիական արդյունաբերությունը, շինարարական նյութերի արտադրության ձեռնարկությունները և այլն:

Արդյունաբերական արտանետումներով բնական միջավայրի աղտոտումը խիստ վնասակար ազդեցություն է գործում մարդկանց, կենդանիների, բույսերի, շենքերի և շինարարությունների վրա, իջեցնում է մթնոլորտի թափանցիկության աստիճանը, բարձրացնում է օդի խոնավությունը, ավելացնում է մշուշոտ օրերի թիվը, նվազեցնում է տեսանելիությունը, առաջացնում է մետաղական իրերի և կառուցվածքների կոռոզիա:

Շրջակա միջավայրի աղտոտում ասելով պետք է հասկանալ միջավայրի հատկությունների (քիմիական, մեխանիկական, ֆիզիկական և նրանց հետ կապված ինֆորմացիոն) փոփոխությունները, որոնք տեղի են ունենում բնական կամ արհեստական գործընթացների հետևանքով և հանգեցնում են միջավայրի ֆունկցիաների վատացմանը յուրաքանչյուր կենսաբանական կամ տեխնոլոգիական օբյեկտի նկատմամբ: Ամերիկացի գիտնական Ֆ. Ռամադը իր "Կիրառական էկոլոգիայի հիմունքներ" գրքում որպես աղտոտման սահմանում նշում է. "Աղտոտումը շրջակա միջավայրի ոչ բարենպաստ փոփոխությունն է, որն ամբողջությամբ կամ մասնավորապես հանդիսանում է մարդու գործունեության արդյունք, ուղիղ կամ անուղղակի ձևով փոխում է էներգիայի բաշխումը, ռադիացիայի մակարդակը, միջավայրի ֆիզիկա-քիմիական հատկությունները և կենդանի օրգանիզմների գոյատևման պայմանները: Հաճախ այդ փոփոխությունները համադրելի են դառնում բնական փոփոխությունների հետ, իսկ մի շարք դեպքերում նույնիսկ գերազանցում են դրանց, ինչի մասին կարելի է դատել աղյուսակ 1-ից:"

**Աղյուսակ 1. Բնական և արդյունաբերական ծագման մի շարք բաղադրիչների արտանետումը մթնոլորտ** **(տոննա/տարի).**

|  |  |  |
| --- | --- | --- |
| Բաղադրիչը | Բնական | Արդյունաբերական |
| Օզոն | 2\*109 | Աննշան |
| Ածխածնի երկօքսիդ CO2 | 7\*1010 | 1,5\*1010 |
| Ածխածնի օքսիդ CO | — | 2\*108 |
| Ծծմբային գազ | 1,42\*108 | 7,3\*107 |
| Ազոտի միացություններ | 1,4\*109 | 1,5\*107 |
| Կախույթային նյութեր | (770-2200)\*106 | (960-2615)\*106 |

Աղտոտման բնական գործընթացները բնության մեջ ունեն իրենց հակոտնյաները, որոնք ընդունակ են չեզոքացնել բնական աղտոտիչի ազդեցությունը, իսկ մարդու կողմից ստեղծված բազմաթիվ նյութեր բնության նկատմամբ օտարածին են: Աղտոտման բնական աղբյուրները սովորաբար հեռու են գտնվում մարդաբնակ վայրերից, իսկ մարդածին աղբյուրները՝ բնակչության կուտակման շրջաններում են:

**6.2 Շրջակա միջավայրի աղտոտման չափերը**

Ըստ չափերի շրջակա միջավայրի աղտոտումը կարելի է բաժանել տեղայինի, տարածաշրջանայինի և գլոբալի: Աղտոտման այս երեք տեսակները սերտորեն կապված են միմյանց հետ: Որպես կանոն առաջնային է համարվում տեղային աղտոտումը, որը, եթե աղտոտման գործընթացի արագությունն ավելի մեծ է, քան մաքրման արագությունը, դառնում է տարածաշրջանային, իսկ հետո քանակական փոփոխությունների կուտակման հետ բերում է շրջակա միջավայրի գլոբալ փոփոխությունների: Գլոբալ աղտոտման համար առավել կարևոր է ժամանակի գործոնը, քանի որ մթնոլորտային օդն օժտված է իր սկզբնական որակները վերականգնելու բնական հատկություննով: Սակայն արտանետումների կտրուկ աճը հանգեցրել է այն բանին, որ աղտոտման չափերը ընդհուպ մոտեցել և հաճախ նույնիսկ գերազանցել են մթնոլորտի ինքնավերականգնման սահմանները:

Աղտոտման ժամանակակից մակարդակների պայմաններում վնասակար նյութերը տարածվում են աղտոտման ազբյուրներից տասնյակ և հարյուրավոր կիլոմետրեր ավելի հեռու: Եվ նույնիսկ աղտոտման աղբյուր հասկացողությունը որոշ չափով փոխում է իր իմաստը: Եթե որևէ արդյունաբերական շրջանում կարելի է առանձնացնել աղտոտման կետային աղբյուրներ, ապա տարածաշրջանի առումով մի ամբողջ արդյունաբերական շրջան կարող է դիտվել որպես միասնական կետային աղբյուր:

Ժամանակակից արդյունաբերությունը նշանակալից ազգեցություն է թողնում բնության վրա գլոբալ չափսերով: Չնայած որ, աղտոտող նյութերի և ջերմային էներգիայի մեծ մասը արտադրվում է սահմանափակ տարածքում, գլխավորապես Հյուսիսային Ամերիկայի, Եվրոպայի և Ասիայի արդյունաբերական շրջաններում, մթնոլորտի շրջանառության և Երկրագնդի ջրային թաղանթի տեղափոխության հետևանքով, որոշ համեմատաբար երկար ապրող աղտոտող նյութերի նշանակալից մասը ցրվում է հսկայական տարածությունների վրա և նույնիսկ ողջ Երկրագունդը ենթարկում տարածաշրջանային և գլոբալ աղտոտման:

Ներկայումս շրջակա միջավայրի մարդածին գլոբալ աղտոտման հետևանքով առաջացել են որոշ կարևոր հիմնահարցեր, որոնց թվին են դասվում՝

* կլիմայի հնարավոր փոփոխություններ՝ կապված տեխնածին ջերմության, ածխաթթու գազի, աէրոզոլային խառնուրդների մթնոլորտ արտանետվելու հետ (ջերմոցային էֆեկտ):
* Երկրագնդի օզոնային շերտի հնարավոր խախտում՝ կապված ֆրեոնների, ազոտի օքսիդների և մի քանի այլ նյութերի մթնոլորտ տրվելու հետ:
* Բնական միջավայրի և կենսոլորտի՝ ռադիոակտիվ նյութերով, ծանր մետաղներով և պեստիցիդներով գլոբալ աղտոտման էկոլոգիական հետևանքները:
* Մթնոլորտային տեղումներով, ջրերի հեսքով, գետերի և ծովային տրանսպորտի կողմից ծովային միջավայրի աղտոտման ընդհանուր հիմնահարցը:
* Աղտոտող նյութերի օդային հոսանքներով տեղափոխումը և թթվային տեղումների հիմնահարցը:

Այդպիսով շրջակա միջավայրի վրա մարդածին ազդեցության չափերը և դրանից բխող վտանգի մակարդակը ստիպում են փնտրել դրանց դեմ պայքարի նոր մոտեցումներ կառավարման և միջպետական համաձայնագրերի հիման վրա:

**6.3 Մթնոլորտի աղտոտման պատճառները և գործոնները**

Մթնոլորտի աղտոտումը կարող է տեղի ունենալ բնական պատճառով՝ երկրաշարժեր, հրաբուխներ, անտառային այրումներ, ջրամբարների կենսաբանական ինքնաղտոտում, մշտապես ընթացող էրոզիա: Սակայն աղտոտում ասելով հասկանում ենք մարդածին աղտոտումները, այսինքն շրջակա միջավայրի աղտոտումը մարդու տնտեսական գործունեության հետևանքով: Ազգաբնակչության և արդյունաբերության հետագա աճը անխուսափելիորեն կբերի աղտոտման վտանգի աճին, որը մեր օրերում առանց այն էլ մեծ չափերի է հասնում:

Հիմնական աղտոտող նյութերը, որոնց պարունակությունը, մթնոլորտում կարգավորվում է ստանդարտների միջոցով, հանդիսանում են ծծմբի երկօքսիդը (SO2), ածխածնի օքսիդները (CO և CO2), ազոտի օքսիդները (NO և NO2), գազային ածխաջրածինները (HC), նաև ծծմբաջրածինը (H2S), ծծմբաածխածինը (CS2), ամոնիակը (NH3),տարբեր հալոգեն պարունակող գազեր:

* *Գազային աղտոտիչներ*

Գոյություն ունի գազային աղտոտիչների 3 հիմնական աղբյուր՝ նյութերի այրում, արդյունաբերական արտադրական գործընթացներ և բնական աղբյուրներ: Վառելիքի այրման արդյունքում նրա ընդհանուր քանակի 78%-ը վեր է ածվում ածխածնի երկօքսիդի, իսկ թերայրման հետևանքով հնարավոր է նաև ածխածնի ենթաօքսիդի առաջացում, որը խիստ թունավոր է: Ածխաջրածինները մթնոլորտ են մտնում վառելանյութերի այրման և նավթամթերքների մշակման ժամանակ, դրանից բացի, շատ ածխաջրածիններ առաջանում են բույսերի աճի և բազմացման ընթացքում: Ըստ գիտնականների գնահատականների, ամբողջ աշխարհում բնական աղբյուրներից ամեն տարի առաջանում է 117 մլն. տոննա ածխաջրածին, իսկ մարդածին աղբյուրներից՝ 100 մլն. տոննա: Սակայն քաղաքների մթնոլորտում առկա ածխաջրածինները հիմնականում իրենցից ներկայացնում են այրման արդյունք:

Ծծոմբի օքսիդների շոշափելի քանակներ մթնոլորտ են արտանետվում սուլֆիդային հանքաքարերից՝ պղնձի, կապարի և ցինկի արտադրության ժամանակ, նաև նավթամթերքների մաքրման ընթացքում: SO2-ի արտանետումների մեծամասնությունը կապված է հնոցներում վառելիքի այրման հետ՝ գործընթացի համար անհևաժեշտ ջերմություն ստանալու համար: Ծծմբի օքսիդներ գոյանում են նաև թղթի և ցելյուլոզայի արտադրության ընթացքում՝ ծծումբ պարունակող նյութերի այրման ընթացքում:

Ածխաջրածիններով մթնոլորտի աղտոտումը քիմիական ձեռնարկությունների, նավթամշակման և մետալուրգիական գործարանների կողմից: Ածխաջրածիններ առաջանում են պլաստմասսաների, ներկերի, սննդային հավելումների, օծանելիքի, ձյութերի, պլաստիֆիկատորների, պիգմենտների, պեստիցիդների արտադրության ժամանակ, նաև կաուչուկների և նավթաքիմիական նյութերի վերամշակման արդյունքում:

Մթնոլորտ արտանետվող քիմիական միացությունների մեջ պարունակվում են մեծ թվով թունավոր նյութեր: Ներկայումս վտանգավոր աղտոտող նյութերի շարքին են դասվում սնդիկի գոլորշիները, վինիլքլորիդը և բենզոլը, որոնց պարունակությունը մթնոլորտում ենթակա է հատուկ վերահսկման:

Հանածո վառելիքի մեծ քանակ ամեն տարի այրվում է կաթսայատների հնոցներում՝ ջերմություն ստանալու համար: Կաթսայատները "ամենակեղտոտ" վառելիքի՝ քարածխի և մազութի, ամենախոշոր սպառողներն են և հանդիսանում են ծծնբի երկօքսիդի և ազոտի օքսիդի արտանետումների հիմնական աղբյուրը:

Գազային աղտոտիչները առաջանում են այրման ընթացքում, իսկ մեխանիկական փոշին կարող է դուրս փչվել քարածուխի բեռնաթափման և կոնվեյերով տեղափոխման, նաև քուրայի մոխրի հեռացման և պահեստավորման ժամանակ: Քարածխի փոշիացումը տեղի է ունենում քամու էրոզիայի պատճառով: Չնայած, որ բնական գազը դիտարկվում է որպես համեմատաբար մաքուր վառելիք, նրա այրման ժամանակ նույնպես գոյանում են աղտոտող նյութեր՝ ազոտի օքսիդներ, ածխածնի օքսիդներ, ածխաջրածիններ, ծուխ:

Մթնոլորտի աղտոտման ոչ պակաս կարևոր աղբյուր է հանդիսանում քաղաքային կոշտ թափոնների այրումը: Բոլոր քաղաքակիրթ երկրներում այդ նպատակի համար գոյություն ունեն աղբայրիչ վառարաններ, արանց կառուցվածքից կախված է արտանետումների պարունակությունը:

Ծխային արտանետումները բաղկացած են այրման համեմատաբար անվնաս գազային նյութերից՝ ածխածնի երկօքսիդից, ջրից, չեզոք ազոտից: Սակայն նրանց ավելցուկը կարող է հանգեցնել մառախուղի գոյացման: Ծխային արտանետումների որսման համար օգտագործվում են զանազան զտիչներ և որսիչներ:

Շրջակա միջավայրի աղտոտումը ներքին այրման շարժիչների արտանետումներով վերջին տարիներին դառնում է մեծ ուշադրության առարկա, մարդու առողջության վրա աճող ապառնալիքի պատճառով:

Տարբեր տրանսպորտային միջոցների, որպես աղտոտման աղբյուրի, համեմատական բնութագիրը բերված է աղյուսակ 2-ում.

աղյուսակ 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Տրանսպորտային միջոց | Ածխաջրեր | Աերոզոլներ | Ծծմբի և ածխածնի օքսիդներ | Ազոտի օքսիդներ |
| Ավտոտրանսպորտ | 6,4 | 1,1 | 0,4 | 6,6 |
| Ինքնաթիռներ | 0,1 | 61,9 | 0,1 | 0,2 |
| Երկաթուղային տրանսպորտ | 0,7 | 0,2 | 0,3 | 0,1 |
| Ծովային տրանսպորտ | 0,5 | 0,6 | 0,3 | 0,2 |

Մթնոլորտի ածխաթթու գազի պարունակության ավելացումը բերում է Երկրագնդի ջերմաստիճանի բարձրացմանը՝ ջերմոցային էֆեկտի առաջացման պատճառով: CO2-ի պարունակության ավելացման դեպքում կարելի է սպասել Երկրագնդի միջին ջերմաստիճանի բարձրացմանը, թեև այդ պարամետրերի միջև եղած կախվածությունը բավականին բարդ է: Այդ հարցի վերաբերյալ կատարվել են բազմաթիվ մոդելային ուսումնասիրություններ: Դատելով դրանց արդյունքներից՝ CO2-ի պարունակության մոտավոր կրկնապատկումը տեղի կունենա 2040 թվականին, որի արդյունքում մոլորակի միջին ջերմաստիճանը կբարձրանա 2 կամ 3 աստիճան: Բևեռային շրջաններում ջերմաստիճանի ավելացումը կարող է գերազանցել տվյալ թիվը մի քանի անգամ: Ջերմաստիճանի փոփոխության վրա մարդածին գործունեության ազդեցության հարցը դեռևս քննարկման փուլում է:

Օդի գոլորշու բազադրությունը նույնպես կարող է փոխվել ջերմաստիճանի փոփոխության ժամանակ: Այդպիսով, մթնոլորտում CO2-ի խտության փոփոխության վերջնական արդյունք կարող են հանդիսանալ բարդ կլիմայական գործընթացները՝ կապված ինչպես ջերմաստիճանի, այնպես էլ տեղումների գոյացման մեխանիզմների փոփոխության հետ:

Ավելի վաղ ենթադրվում էր, որ տրանսպորտային ավիացիայի կողմից արտանետվող ազոտի օքսիդները հանդիսանում են օզոնային շերտի քայքայման հիմնական պատճառը: Սակայն քանակական չափումները ցույց տվեցին, որ այդ աղբյուրները չնչին են ֆրեոնների և նրա նման նյութերի օզոնային շերտի վրա բացասական ազդեցության առումով: Նրանք ունեն արհեստական ծագում և լայնորեն կիրառվում են սառեցնող և տարբեր աէրոզոլային սարքերում: Մթնոլորտում օզոնը կազմում է շատ փոքրիկ չափաբաժին՝ ըստ ծավալի և զանգվածի մեկ միլիոներորդ մասից պակաս: Նրա հիմնական մասը կենտրոնացված է ստրատոսֆերայում՝ մինչև 90%-ը: Մնացած 10%-ը կուտակված է մթնոլորտի ներքին շերտերում: Այստեղ արդեն օզոնը հանդիսանում է օդի չափազանց վտանգավոր աղտոտիչ: Նա ազդում է շնչուղիների վրա, գրգռում է աչքերը, խախտում է բուսականության աճը և այլն, շնչառության համար օգտագործվող օդում նրա խտությունը չպիտի գերազանցի 150-200 մկգ/մ3:

Օզոնը գոյանում է էլեկտրական լիցքաթափման արդյունքում, սակայն շատ ավելի կարևոր են ֆոտոքիմիական ռեակցիաները միջնորդ նյութերի (ազոտի օքսիդների կամ ածխաջրածինների) մասնակցությամբ: Մեծ քաղաքներում արդյունաբերական և ավտոմոբիլային արտանետումների արդյունքում, որոնք փոխազդում են մեկը մյուսի և այլ գազերի հետ, գոյանում են բարդ քիմիական միացություններ, առաջանում է ֆոտոքիմիական մշուշ, որն ունի օզոնի մեծ խտություն:

Մթնոլորտի աղտոտման ժամանակակից վիճակի և դինամիակայի վերլուծությունը ցույց է տալիս, որ չնայած դանդաղորեն, բայց տեղի ՝ ունենում բեռնվածության նվազում քաղաքի օդային ավազանի վրա՝ գազի ու փոշի որսման մաքրող սարքերի տեղադրման հաշվին: Մի քանի տարիների ընթացքում դեպի մթնոլորտ արտանետումների քանակը պակասել է արտադրության կրճատման հաշվին:

**6.4 Շրջակա միջավայրի աղտոտվածության դեմ պայքարի հիմնական ուղիները**

Մմթնոլորտի աղտոտման չափերի օրեցօր ավելացումը պահանջում է աղտոտումից նրա պաշտպանության և կանխարգելման արագ և էֆեկտիվ ձևերի ու մեթոդների մշակում: Մթնոլորտը կարող է պարունակել աղտոտիչի որոշակի քանակ, որը վնասակար ներգործություն չի դրսևորում մարդու օրգանիզմի վրա և շրջակա միջավայրի վրա, քանի որ տեղի է ունենում դրանց մաքրումը բնական ճանապարհով: Մթնոլորտի և, ընդհանրապես, կենսոլորտի պաշտպանության ամենալավ մեթոդը անթափոն կամ քիչ թափոններով արտադրական տեխնոլոգիաների ներդրումն է: Սակայն արդյունաբերական ձեռնարկությունների մեծ մասը դեռևս կիրառում են բաց տեխնոլոգիական պրոցեսներ, որոնք մթնոլորտ են արտանետում պինդ մասնիկներ, փոշիներ և թափոնային գազեր:

Մթնոլորտի աղտոտումից պաշտպանության միջոցառումների համալիրն ընդգրկում է՝

* արտադրությունների տեխնոլոգիաների կատարելագործումը՝ վտանգավոր արտադրումները կրճատելու նպատակով;
* առավել վտանգավոր արտադրությունների կրճատում;
* արտադրական ձեռնարկությունների շուրջը սանիտարական գոտինեչի ստեղծում;
* մթնոլորտ արտանետվող քիմիական միացությունների ցրում;
* արտադրություններում առավել արդյունավետ մաքրման սարքավորումների կիրառում;
* կանաչ տարածքների ստեղծում, որոնք օժտված են փոշեկլանիչ և գազակլանիչ հատկություններով:

Որպես սանիտարական գոտի ընդունված է արտադրական տարածքների միջև ընկած տարածությունը, որտեղ վտանգավոր նյութերի խտությունները չպետք է գերազանցեն ՍԹԽ (սահմանային թույլատրելի խտությունը) և հատկապես դրանց սահմանակից բնակավայրերի մթնոլորտային օդում մշտապես պետք է իրականացնել վտանգավոր նյութերի ՍԹԽ-ների հսկում: Սահմանային թույլատրելի խտությունը տվյալ նյութի այն սահմանային խտությունն է, որից բարձր արժեքի դեպքում նյութը համարվում է վտանգավոր մարդու առողջության և շրջակա կենդանական և բուսական աշխարհի համար:

Արտադրական ձեռնարկությունները, կախված շրջակա միջավայրի համար դրանց վտանգավորության աստիճանից, ըստ սանիտարահիգիենիկ ցուցանիշների բաժանվում են 5 դասի: Առավել վտանգավոր է առաջին դասը և պակաս վտանգավոր՝ հինգերորդը: Արտադրական ձեռնարկության յուրաքանչյուր դասին համապատասխանում է իր սանիտարական գոտին. առաջին դասի ձեռնարկությունների համար այն կազմում է 1000մ, երկրորդ դասի համար՝ 500մ, երրորդ դասի համար՝ 300մ, չորրորդի համար՝ 100մ, և հինգերորդը՝ 50մ: Որպես կանոն արտադրություններում գազերի և փոշիների համար օգտագործում են 200-350մ բարձրությամբ խողովակներ: Բարձր խողովակների օգտագործումը հնարավորություն է տալիս աղտոտող նյութը բաց թողնել բաց տարածության վրա, որը նոսրացնում է այդ նյութի խտությունը: Օրինակ, 200մ բաձրությամբ փոշեգազային խողովակները արտանետում են 20կմ հեռավորոթյան վրա, իսկ 250մ-ը՝ 75կմ-ի վրա:

Արտանետումների ծավալի նվազեցմաան կամ կանխման նմատակով, որպես կանոն, արտադրական ձեռնարկություններում իրականացվում են գազամաքրման կամ փոշին որսալու գործընթացներ, կամ էլ օգտագործվում են հաուկ սարքեր: Արդյունքում աղտոտիչների քիմիական կամ ֆիզիկական հատկությունների փոփոխման հետևանքով դրանք դառնում են պակաս վտանգավոր:

Մթնոլորտ արտանետվող գազերը փոշուցմաքրելու համար արտադրական ձեռնարկություններում կիրառում են բազմաթիվ սարքավորումներ, որոնք կարելի է դասակարգել հետևյալ կերպ.

* մեխանիկական փոշեկլանիչ սարքեր, որոնցում փոշին առանձնանում է գազից գրավիտացիոն, իներցիոն կամ կենտրոնախույզ ուժի ազդեցության տակ;
* էլեկտրազտիչներ, որոնցում փոշու նստեցումը իրականացվում է գազի իոնացմամբ և փոշեհատիկներին էլեկտրական լիցք հաղորդելու հաշվին;
* հիդրավլիկ սարքեր, որոնցում պինդ մասնիկները կլանվում են հեղուկով (թաց եղանակ);
* ծակոտկեն զտիչներ, որոնք կլանում են մանր դիսպերսային փոշին:

Արտանետվող գազերը վտանգավոր քիմիական թափոնային գազերից և գոլորշիներից մաքրելու համար կիրառվում են գազամաքրմանբազմաթիվ եղանակներ, որոնք կարելի է դասակարգել հետևյալ կերպ.

* աբսորբցիա՝ գազերի կլանումը հեղուկով լվանալու միջոցով ( ծավալային սորբցիա);
* ադսորբցիա՝ գազերի կլանումը պինդ մասնիկներով (մակերեսային սորբցիա);
* օքսիդավերականգնման պրոցեսներ (բարձր ջերմաստիճանային և կատալիտիկ), որի արդյունքում առաջանում են նոր, էկոլոգիապես ավելի պակաս վտանգավոր նյութեր:

Գազամաքրման համար կիրառվող այս եղանակները թույլ են տալիս բարելավել էկոլոգիական իրավիճակը, սակայն ամբողջությամբ չեն լուծում ձեռնարկությունների էկոլոգիական խնդիրները: Անկասկած առավել արդյունավետ եղանակ է անթափոն կամ քիչ թափոնային արտադրական տեխնոլոգիաների կիրառումը:

**Եզրակացություն**

Այսպիսով ի՞նչ եզրակացություններ կարող ենք անել քննարկված աշխատանքի վերաբերյալ:

* Ստեղծվել է համակարգ, որը կարող է հայտնաբերել տարբեր տեսակի օբյեկտներ (մասնավորապես ավտոմեքենաներ և մոտոցիկլետներ):
* Դասակարգել հայտնաբերված օբյեկտները ըստ դասերի:
* Հաշվարկել ամեն դասին պատկանող օբյեկտների քանակը:

Խնդրի լուծման համար մշակվել են օպտիմալ և արագագործ ալգորիթմներ՝

* Լոկալ հետաքրքրության տարածքի սահմանում:
* Օբյեկտների հաշվարկման համար նախատեսված սահմանագծերի սահմանում:

**Գրականություն**

1. Троелсен Э. - Язык программирования C# 2010 и платформа .NET 4.0
2. Нейгел К. - C# 2008 и платформа .NET 3.5 для профессионалов - 2008
3. [Matthew MacDonald](http://www.apress.com/author/author/view/id/1836) - Pro WPF in C# 2010 Windows Presentation Foundation in .NET 4 , 3rd edition
4. Laganiere Robert - OpenCV 2 Computer Vision Application Programming Cookbook – 2011
5. Gary Bradski & Adrian Kaebler – LearningOpenCV - 2008
6. Дэвид Форсайт, Жан Понс - Компьютерное зрение. Современний подход - 2004
7. Линда Шапиро, Джордж Стокман - Компьютерное зрение - 2006
8. Paul Toft Duizer, Dennis Mølholm Hansen - Multi-View Video Surveillance of Outdoor Traffic – 2007

**Հավելված**

**mainForm.Designer.cs**

namespace SRTS

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.inputImageBox = new Emgu.CV.UI.ImageBox();

this.outputImageBox = new Emgu.CV.UI.ImageBox();

this.inputLabel = new System.Windows.Forms.Label();

this.outputLabel = new System.Windows.Forms.Label();

this.startButton = new System.Windows.Forms.Button();

this.camera2ImageBox = new Emgu.CV.UI.ImageBox();

this.camera3ImageBox = new Emgu.CV.UI.ImageBox();

this.camera4ImageBox = new Emgu.CV.UI.ImageBox();

this.camera1ImageBox = new Emgu.CV.UI.ImageBox();

this.optionsButton = new System.Windows.Forms.Button();

this.crossroadNameLabel = new System.Windows.Forms.Label();

this.camera1Label = new System.Windows.Forms.Label();

this.camera2Label = new System.Windows.Forms.Label();

this.camera3Label = new System.Windows.Forms.Label();

this.camera4Label = new System.Windows.Forms.Label();

this.stopButton = new System.Windows.Forms.Button();

this.mainTimer = new System.Windows.Forms.Timer(this.components);

this.trackBar1 = new System.Windows.Forms.TrackBar();

this.carCountLabel = new System.Windows.Forms.Label();

this.motorcyclesCountLabel = new System.Windows.Forms.Label();

this.carsCountLbl = new System.Windows.Forms.Label();

this.motorcyclesCountLbl = new System.Windows.Forms.Label();

this.showSeperationLinesCheckBox = new System.Windows.Forms.CheckBox();

((System.ComponentModel.ISupportInitialize)(this.inputImageBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.outputImageBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.camera2ImageBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.camera3ImageBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.camera4ImageBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.camera1ImageBox)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).BeginInit();

this.SuspendLayout();

//

// inputImageBox

//

this.inputImageBox.FunctionalMode = Emgu.CV.UI.ImageBox.FunctionalModeOption.RightClickMenu;

this.inputImageBox.Location = new System.Drawing.Point(234, 113);

this.inputImageBox.Name = "inputImageBox";

this.inputImageBox.Size = new System.Drawing.Size(300, 300);

this.inputImageBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

this.inputImageBox.TabIndex = 2;

this.inputImageBox.TabStop = false;

//

// outputImageBox

//

this.outputImageBox.FunctionalMode = Emgu.CV.UI.ImageBox.FunctionalModeOption.RightClickMenu;

this.outputImageBox.Location = new System.Drawing.Point(701, 113);

this.outputImageBox.Name = "outputImageBox";

this.outputImageBox.Size = new System.Drawing.Size(300, 300);

this.outputImageBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;

this.outputImageBox.TabIndex = 2;

this.outputImageBox.TabStop = false;

this.outputImageBox.Visible = false;

//

// inputLabel

//

this.inputLabel.AutoSize = true;

this.inputLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 16F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.inputLabel.Location = new System.Drawing.Point(229, 73);

this.inputLabel.Name = "inputLabel";

this.inputLabel.Size = new System.Drawing.Size(60, 26);

this.inputLabel.TabIndex = 3;

this.inputLabel.Text = "Input";

//

// outputLabel

//

this.outputLabel.AutoSize = true;

this.outputLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 16F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.outputLabel.Location = new System.Drawing.Point(696, 73);

this.outputLabel.Name = "outputLabel";

this.outputLabel.Size = new System.Drawing.Size(77, 26);

this.outputLabel.TabIndex = 4;

this.outputLabel.Text = "Output";

this.outputLabel.Visible = false;

//

// startButton

//

this.startButton.Location = new System.Drawing.Point(1059, 160);

this.startButton.Name = "startButton";

this.startButton.Size = new System.Drawing.Size(148, 38);

this.startButton.TabIndex = 5;

this.startButton.Text = "Start";

this.startButton.UseVisualStyleBackColor = true;

this.startButton.Click += new System.EventHandler(this.startButton\_Click);

//

// camera2ImageBox

//

this.camera2ImageBox.FunctionalMode = Emgu.CV.UI.ImageBox.FunctionalModeOption.Minimum;

this.camera2ImageBox.Location = new System.Drawing.Point(12, 240);

this.camera2ImageBox.Name = "camera2ImageBox";

this.camera2ImageBox.Size = new System.Drawing.Size(150, 150);

this.camera2ImageBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.camera2ImageBox.TabIndex = 6;

this.camera2ImageBox.TabStop = false;

this.camera2ImageBox.Click += new System.EventHandler(this.cameraImageBox\_Click);

//

// camera3ImageBox

//

this.camera3ImageBox.FunctionalMode = Emgu.CV.UI.ImageBox.FunctionalModeOption.Minimum;

this.camera3ImageBox.Location = new System.Drawing.Point(12, 430);

this.camera3ImageBox.Name = "camera3ImageBox";

this.camera3ImageBox.Size = new System.Drawing.Size(150, 150);

this.camera3ImageBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.camera3ImageBox.TabIndex = 7;

this.camera3ImageBox.TabStop = false;

this.camera3ImageBox.Click += new System.EventHandler(this.cameraImageBox\_Click);

//

// camera4ImageBox

//

this.camera4ImageBox.FunctionalMode = Emgu.CV.UI.ImageBox.FunctionalModeOption.Minimum;

this.camera4ImageBox.Location = new System.Drawing.Point(12, 620);

this.camera4ImageBox.Name = "camera4ImageBox";

this.camera4ImageBox.Size = new System.Drawing.Size(150, 150);

this.camera4ImageBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.camera4ImageBox.TabIndex = 8;

this.camera4ImageBox.TabStop = false;

this.camera4ImageBox.Click += new System.EventHandler(this.cameraImageBox\_Click);

//

// camera1ImageBox

//

this.camera1ImageBox.FunctionalMode = Emgu.CV.UI.ImageBox.FunctionalModeOption.Minimum;

this.camera1ImageBox.Location = new System.Drawing.Point(12, 50);

this.camera1ImageBox.Name = "camera1ImageBox";

this.camera1ImageBox.Size = new System.Drawing.Size(150, 150);

this.camera1ImageBox.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.camera1ImageBox.TabIndex = 9;

this.camera1ImageBox.TabStop = false;

this.camera1ImageBox.Click += new System.EventHandler(this.cameraImageBox\_Click);

//

// optionsButton

//

this.optionsButton.Location = new System.Drawing.Point(1059, 113);

this.optionsButton.Name = "optionsButton";

this.optionsButton.Size = new System.Drawing.Size(148, 38);

this.optionsButton.TabIndex = 10;

this.optionsButton.Text = "Options";

this.optionsButton.UseVisualStyleBackColor = true;

this.optionsButton.Click += new System.EventHandler(this.optionsButton\_Click);

//

// crossroadNameLabel

//

this.crossroadNameLabel.AutoSize = true;

this.crossroadNameLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 20F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.crossroadNameLabel.Location = new System.Drawing.Point(539, 9);

this.crossroadNameLabel.Name = "crossroadNameLabel";

this.crossroadNameLabel.Size = new System.Drawing.Size(260, 31);

this.crossroadNameLabel.TabIndex = 11;

this.crossroadNameLabel.Text = "Costa Rica Freeway";

this.crossroadNameLabel.TextAlign = System.Drawing.ContentAlignment.TopCenter;

//

// camera1Label

//

this.camera1Label.AutoSize = true;

this.camera1Label.Font = new System.Drawing.Font("Microsoft Sans Serif", 11F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.camera1Label.Location = new System.Drawing.Point(51, 29);

this.camera1Label.Name = "camera1Label";

this.camera1Label.Size = new System.Drawing.Size(73, 18);

this.camera1Label.TabIndex = 12;

this.camera1Label.Text = "Camera 1";

//

// camera2Label

//

this.camera2Label.AutoSize = true;

this.camera2Label.Font = new System.Drawing.Font("Microsoft Sans Serif", 11F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.camera2Label.Location = new System.Drawing.Point(51, 219);

this.camera2Label.Name = "camera2Label";

this.camera2Label.Size = new System.Drawing.Size(73, 18);

this.camera2Label.TabIndex = 13;

this.camera2Label.Text = "Camera 2";

//

// camera3Label

//

this.camera3Label.AutoSize = true;

this.camera3Label.Font = new System.Drawing.Font("Microsoft Sans Serif", 11F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.camera3Label.Location = new System.Drawing.Point(51, 409);

this.camera3Label.Name = "camera3Label";

this.camera3Label.Size = new System.Drawing.Size(73, 18);

this.camera3Label.TabIndex = 14;

this.camera3Label.Text = "Camera 3";

//

// camera4Label

//

this.camera4Label.AutoSize = true;

this.camera4Label.Font = new System.Drawing.Font("Microsoft Sans Serif", 11F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.camera4Label.Location = new System.Drawing.Point(51, 599);

this.camera4Label.Name = "camera4Label";

this.camera4Label.Size = new System.Drawing.Size(73, 18);

this.camera4Label.TabIndex = 15;

this.camera4Label.Text = "Camera 4";

//

// stopButton

//

this.stopButton.Location = new System.Drawing.Point(1059, 205);

this.stopButton.Name = "stopButton";

this.stopButton.Size = new System.Drawing.Size(148, 38);

this.stopButton.TabIndex = 16;

this.stopButton.Text = "Stop";

this.stopButton.UseVisualStyleBackColor = true;

this.stopButton.Click += new System.EventHandler(this.stopButton\_Click);

//

// mainTimer

//

this.mainTimer.Tick += new System.EventHandler(this.mainTimer\_Tick);

//

// trackBar1

//

this.trackBar1.Location = new System.Drawing.Point(234, 486);

this.trackBar1.Maximum = 255;

this.trackBar1.Name = "trackBar1";

this.trackBar1.Size = new System.Drawing.Size(104, 45);

this.trackBar1.TabIndex = 17;

this.trackBar1.Value = 83;

this.trackBar1.Visible = false;

//

// carCountLabel

//

this.carCountLabel.AutoSize = true;

this.carCountLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.carCountLabel.Location = new System.Drawing.Point(1055, 253);

this.carCountLabel.Name = "carCountLabel";

this.carCountLabel.Size = new System.Drawing.Size(89, 20);

this.carCountLabel.TabIndex = 18;

this.carCountLabel.Text = "Cars Count";

this.carCountLabel.TextAlign = System.Drawing.ContentAlignment.TopCenter;

//

// motorcyclesCountLabel

//

this.motorcyclesCountLabel.AutoSize = true;

this.motorcyclesCountLabel.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.motorcyclesCountLabel.Location = new System.Drawing.Point(1055, 335);

this.motorcyclesCountLabel.Name = "motorcyclesCountLabel";

this.motorcyclesCountLabel.Size = new System.Drawing.Size(140, 20);

this.motorcyclesCountLabel.TabIndex = 19;

this.motorcyclesCountLabel.Text = "Motorcycles Count";

this.motorcyclesCountLabel.TextAlign = System.Drawing.ContentAlignment.TopCenter;

//

// carsCountLbl

//

this.carsCountLbl.AutoSize = true;

this.carsCountLbl.Font = new System.Drawing.Font("Microsoft Sans Serif", 18F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.carsCountLbl.Location = new System.Drawing.Point(1055, 292);

this.carsCountLbl.Name = "carsCountLbl";

this.carsCountLbl.Size = new System.Drawing.Size(26, 29);

this.carsCountLbl.TabIndex = 20;

this.carsCountLbl.Text = "0";

this.carsCountLbl.TextAlign = System.Drawing.ContentAlignment.TopCenter;

//

// motorcyclesCountLbl

//

this.motorcyclesCountLbl.AutoSize = true;

this.motorcyclesCountLbl.Font = new System.Drawing.Font("Microsoft Sans Serif", 18F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.motorcyclesCountLbl.Location = new System.Drawing.Point(1055, 374);

this.motorcyclesCountLbl.Name = "motorcyclesCountLbl";

this.motorcyclesCountLbl.Size = new System.Drawing.Size(26, 29);

this.motorcyclesCountLbl.TabIndex = 21;

this.motorcyclesCountLbl.Text = "0";

this.motorcyclesCountLbl.TextAlign = System.Drawing.ContentAlignment.TopCenter;

//

// showSeperationLinesCheckBox

//

this.showSeperationLinesCheckBox.AutoSize = true;

this.showSeperationLinesCheckBox.Location = new System.Drawing.Point(1059, 412);

this.showSeperationLinesCheckBox.Name = "showSeperationLinesCheckBox";

this.showSeperationLinesCheckBox.Size = new System.Drawing.Size(135, 17);

this.showSeperationLinesCheckBox.TabIndex = 22;

this.showSeperationLinesCheckBox.Text = "Show Separation Lines";

this.showSeperationLinesCheckBox.UseVisualStyleBackColor = true;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1233, 786);

this.Controls.Add(this.showSeperationLinesCheckBox);

this.Controls.Add(this.motorcyclesCountLbl);

this.Controls.Add(this.carsCountLbl);

this.Controls.Add(this.motorcyclesCountLabel);

this.Controls.Add(this.carCountLabel);

this.Controls.Add(this.trackBar1);

this.Controls.Add(this.stopButton);

this.Controls.Add(this.camera4Label);

this.Controls.Add(this.camera3Label);

this.Controls.Add(this.camera2Label);

this.Controls.Add(this.camera1Label);

this.Controls.Add(this.crossroadNameLabel);

this.Controls.Add(this.optionsButton);

this.Controls.Add(this.camera1ImageBox);

this.Controls.Add(this.camera4ImageBox);

this.Controls.Add(this.camera3ImageBox);

this.Controls.Add(this.camera2ImageBox);

this.Controls.Add(this.startButton);

this.Controls.Add(this.outputLabel);

this.Controls.Add(this.inputLabel);

this.Controls.Add(this.outputImageBox);

this.Controls.Add(this.inputImageBox);

this.Name = "Form1";

this.Text = "SRTS Smart Recognition & Tracking System";

((System.ComponentModel.ISupportInitialize)(this.inputImageBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.outputImageBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.camera2ImageBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.camera3ImageBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.camera4ImageBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.camera1ImageBox)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private Emgu.CV.UI.ImageBox inputImageBox;

private Emgu.CV.UI.ImageBox outputImageBox;

private System.Windows.Forms.Label inputLabel;

private System.Windows.Forms.Label outputLabel;

private System.Windows.Forms.Button startButton;

private Emgu.CV.UI.ImageBox camera2ImageBox;

private Emgu.CV.UI.ImageBox camera3ImageBox;

private Emgu.CV.UI.ImageBox camera4ImageBox;

private Emgu.CV.UI.ImageBox camera1ImageBox;

private System.Windows.Forms.Button optionsButton;

private System.Windows.Forms.Label crossroadNameLabel;

private System.Windows.Forms.Label camera1Label;

private System.Windows.Forms.Label camera2Label;

private System.Windows.Forms.Label camera3Label;

private System.Windows.Forms.Label camera4Label;

private System.Windows.Forms.Button stopButton;

private System.Windows.Forms.Timer mainTimer;

private System.Windows.Forms.TrackBar trackBar1;

private System.Windows.Forms.Label carCountLabel;

private System.Windows.Forms.Label motorcyclesCountLabel;

private System.Windows.Forms.Label carsCountLbl;

private System.Windows.Forms.Label motorcyclesCountLbl;

private System.Windows.Forms.CheckBox showSeperationLinesCheckBox;

}

}

**mainForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

#region Using Emgu Dll-s

//Using Emgu Dll-s

using Emgu.CV;

using Emgu.Util;

using Emgu.CV.CvEnum;

using Emgu.CV.Structure;

using Emgu.CV.UI;

using Emgu.CV.Util;

using Emgu.CV.VideoSurveillance;

#endregion

namespace SRTS

{

public partial class mainForm : Form

{

Image<Bgr, byte> input = new Image<Bgr, byte>(@"C:\Users\HM\Desktop\a.jpg");

Image<Bgr, byte> inputFrame1, inputFrame2, backgroundFrame, currentCar = new Image<Bgr, byte>(@"C:\Users\HM\Desktop\currentCar.jpg");

Image<Gray, byte> inputGrayFrame1, inputGrayFrame2, backgroundGrayFrame;

Image<Bgr, byte> drawingLayerImage;

Point lineStartPoint, lineEndPoint;

float imageSizeChangingCoefficient = 0.3f;

int count, currentCarNumber;

double max;

bool startCount, k;

List<LineSegment2D> linesToDraw = new List<LineSegment2D>();

Capture capture = new Capture(@"C:\Users\HM\Desktop\Costa Rica Freeway Traffic.mp4");

public mainForm()

{

InitializeComponent();

//crossroadNameLabel.Text = "Sayat-Nova--Khanjyan";

inputFrame1 = capture.QuerySmallFrame();

inputFrame2 = capture.QuerySmallFrame();

drawingLayerImage = inputFrame2;

mainTimer.Interval = (int)((1 / capture.GetCaptureProperty(CAP\_PROP.CV\_CAP\_PROP\_FPS)) \* 1000);

}

private void mainTimer\_Tick(object sender, EventArgs e)

{

if (capture.GetCaptureProperty(CAP\_PROP.CV\_CAP\_PROP\_POS\_FRAMES) >= capture.GetCaptureProperty(CAP\_PROP.CV\_CAP\_PROP\_FRAME\_COUNT) - 2)

{

capture.SetCaptureProperty(CAP\_PROP.CV\_CAP\_PROP\_POS\_FRAMES, 0);

//MessageBox.Show(max.ToString());

}

inputFrame1 = capture.QuerySmallFrame();

inputFrame2 = capture.QuerySmallFrame();

drawingLayerImage = inputFrame2.Clone();

if(showSeperationLinesCheckBox.Checked)

foreach (var line in linesToDraw)

inputFrame2.Draw(line, new Bgr(0, 0, 255), 1);

inputImageBox.Height = (int)(inputFrame1.Height / imageSizeChangingCoefficient);

inputImageBox.Width = (int)(inputFrame1.Width / imageSizeChangingCoefficient);

outputImageBox.Height = (int)(inputFrame1.Height / imageSizeChangingCoefficient);

outputImageBox.Width = (int)(inputFrame1.Width / imageSizeChangingCoefficient);

camera1ImageBox.Image = inputFrame2;

camera2ImageBox.Image = inputFrame2;

camera3ImageBox.Image = inputFrame2;

camera4ImageBox.Image = inputFrame2;

/\*

camera1ImageBox.Image = capture.QuerySmallFrame();

camera2ImageBox.Image = capture2.QuerySmallFrame();

camera3ImageBox.Image = capture3.QuerySmallFrame();

camera4ImageBox.Image = capture4.QuerySmallFrame();

\*/

inputImageBox.Image = inputFrame2;

outputImageBox.Image = Process();

carsCountLbl.Text = count.ToString();

}

private void cameraImageBox\_Click(object sender, EventArgs e)

{

inputImageBox.Image = (sender as ImageBox).Image;

}

private void stopButton\_Click(object sender, EventArgs e)

{

mainTimer.Enabled = false;

backgroundFrame = inputFrame2.Clone();

}

private void startButton\_Click(object sender, EventArgs e)

{

mainTimer.Enabled = true;

}

private Image<Gray, byte> Process()

{

inputGrayFrame1 = inputFrame1.Convert<Gray, byte>();

inputGrayFrame2 = inputFrame2.Convert<Gray, byte>();

backgroundGrayFrame = backgroundFrame.Convert<Gray, byte>();

//Image<Bgr, byte> processedImage = inputFrame2.Sub(inputFrame1);

Image<Gray, byte> processedGrayImage = inputGrayFrame2.Sub(inputGrayFrame1);

//processedImage = processedImage.ThresholdBinary(new Bgr(10, 10, 10), new Bgr(255, 255, 255));

processedGrayImage = processedGrayImage.ThresholdBinary(new Gray(5), new Gray(255));

processedGrayImage = processedGrayImage.Or(inputGrayFrame2.ThresholdBinaryInv(new Gray(trackBar1.Value), new Gray(255)));

//processedGrayImage = processedGrayImage.Or(inputGrayFrame2.Sub(backgroundGrayFrame).ThresholdBinary(new Gray(trackBar1.Value), new Gray(255)));

processedGrayImage = objectSeperation2(processedGrayImage);

return processedGrayImage;

}

private Image<Gray, byte> objectSeperation(Image<Gray, byte> inputImage)

{

int minRectDim = 10;

int maxRectDim = 50;

Image<Gray, byte> outputImage = inputImage.Clone();

for (int y = 0; y < inputImage.Rows; y++)

{

for (int x = 0; x < inputImage.Cols; x++)

{

if (inputImage[y, x].Intensity == 255)

{

Rectangle rect;

MCvConnectedComp conRect = new MCvConnectedComp();

//MCvConnectedComp conRect2 = new MCvConnectedComp();

CvInvoke.cvFloodFill(inputImage.Ptr, new Point(x, y), new MCvScalar(200), new MCvScalar(0), new MCvScalar(0), out conRect, CONNECTIVITY.EIGHT\_CONNECTED, FLOODFILL\_FLAG.DEFAULT, IntPtr.Zero);

rect = conRect.rect;

if (rect.Width >= minRectDim && rect.Width <= maxRectDim && rect.Height >= minRectDim && rect.Height <= maxRectDim)

{

Point p = new Point(rect.Location.X + rect.Width / 2, rect.Location.Y + rect.Height / 2);

//CvInvoke.cvFloodFill(img6.Ptr, p, new MCvScalar(0), new MCvScalar(2), new MCvScalar(2), out conRect2, CONNECTIVITY.EIGHT\_CONNECTED, FLOODFILL\_FLAG.DEFAULT, IntPtr.Zero);

int xCord = rect.X + rect.Width / 2;

int yCord = rect.Y + rect.Height / 2;

outputImage.Draw(rect, new Gray(254), 2);

inputFrame2.Draw(rect, new Bgr(0, 255, 0), 2);

}

}

}

}

return outputImage;

}

private Image<Gray, byte> objectSeperation2(Image<Gray, byte> inputImage)

{

int minRectDim = 10;

int maxRectDim = 50;

Image<Gray, byte> outputImage = inputImage.Clone();

int objectsCount = 0;

/\*if(k)

MatchingCarTemplate(currentCar);\*/

foreach (var line in linesToDraw)

{

for (int y = line.P1.Y; y < line.P2.Y; y++)

{

for (int x = line.P1.X; x < line.P2.X; x++)

{

if (inputImage[y, x].Intensity == 255)

{

Rectangle rect;

MCvConnectedComp conRect = new MCvConnectedComp();

//MCvConnectedComp conRect2 = new MCvConnectedComp();

CvInvoke.cvFloodFill(inputImage.Ptr, new Point(x, y), new MCvScalar(50 + currentCarNumber), new MCvScalar(0), new MCvScalar(0), out conRect, CONNECTIVITY.EIGHT\_CONNECTED, FLOODFILL\_FLAG.DEFAULT, IntPtr.Zero);

rect = conRect.rect;

if (rect.Width >= minRectDim && rect.Width <= maxRectDim && rect.Height >= minRectDim && rect.Height <= maxRectDim)

{

/\* if (!k)

{

currentCar = new Image<Bgr, byte>(rect.Size);

currentCar = inputFrame2.GetSubRect(rect);

//currentCar.Save(@"C:\Users\HM\Desktop\currentCar.jpg");

k = true;

}\*/

//CvInvoke.cvDrawContours(inputFrame2.Ptr, conRect.contour, new MCvScalar(255, 0, 0), new MCvScalar(255, 255, 255), 1, 10, LINE\_TYPE.EIGHT\_CONNECTED, new Point(0));

Point p = new Point(rect.Location.X + rect.Width / 2, rect.Location.Y + rect.Height / 2);

//CvInvoke.cvFloodFill(img6.Ptr, p, new MCvScalar(0), new MCvScalar(2), new MCvScalar(2), out conRect2, CONNECTIVITY.EIGHT\_CONNECTED, FLOODFILL\_FLAG.DEFAULT, IntPtr.Zero);

int xCord = rect.X + rect.Width / 2;

int yCord = rect.Y + rect.Height / 2;

outputImage.Draw(rect, new Gray(254), 1);

inputFrame2.Draw(rect, new Bgr(0, 255, 0), 1);

objectsCount++;

}

}

}

}

}

if (objectsCount > 0 && startCount != false)

{

count += objectsCount;

startCount = false;

}

if (objectsCount == 0)

startCount = true;

return outputImage;

}

private void MatchingCarTemplate(Image<Bgr, byte> currentCar)

{

Image<Gray, float> templateResult = new Image<Gray, float>(new Size(inputFrame2.Width - currentCar.Width + 1, inputFrame2.Height - currentCar.Height + 1));

CvInvoke.cvMatchTemplate(inputFrame2.Ptr, currentCar.Ptr, templateResult.Ptr, TM\_TYPE.CV\_TM\_CCOEFF);

double minVal = 0, maxVal = 0;

Point minLoc = new Point(), maxLoc = new Point();

CvInvoke.cvMinMaxLoc(templateResult.Ptr, ref minVal, ref maxVal, ref minLoc, ref maxLoc, IntPtr.Zero);

if (maxVal > max)

max = maxVal;

if (maxVal > 0.41)

inputFrame2.Draw(new Rectangle(maxLoc, currentCar.Size), new Bgr(255, 0, 0), 1);

camera1ImageBox.Image = templateResult;

camera2ImageBox.Image = currentCar;

}

private void optionsButton\_Click(object sender, EventArgs e)

{

Options options = new Options();

if (options.ShowDialog() == System.Windows.Forms.DialogResult.Cancel)

{

capture = options.capture;

linesToDraw = options.linesToDraw;

backgroundFrame = options.backgroundFrame;

}

}

}

}