VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

Magistro baigiamasis darbas

Automatizuoti vaizdų apdorojimo algoritmai

Atliko:
Monika Kelpšaitė parašas

Vadovas:
dr. Tadas Meškauskas

Vilnius
2021

Turinys

Sutartinis terminų žodynas

- *DIP - druskos ir pipirų nuotraukos užtriukšminimo tipas.*
- *MPN - maksimaliai pasuktų nuotraukų skaičius. Parametras naudojamas tiesinimo modelio (aprašytas skyrelyje*
- *ResNet - „Deep Residual Network“, gilus išliekamasis neuroninis tinklas.*
- *RotNet - nuotraukos rotacijai atstatyti skirtas modelis, naudojantis ResNet50 tinklą*
- *TTK - teisingas tiesinimo kampas, nurodantis kokiu kampu testuojama nuotrauka turi būti pasukta, kad būtų ištiesinta.*

Santrauka

Šiame magistro baigiamajame darbe analizuojami trys kraštų aptikimo algoritmai: „Sobel“, „Scharr“ ir „Canny“. Darbe išsamiai analizuojama, kaip algoritmai veikia su realaus pasaulio nuotraukomis, atspindinčiomis realaus pasaulio vaizdus, be to, kaip algoritmai susidoroja, kuomet yra pridėtas triukšmas, ir kokį poveikį kraštų radimui daro didesnis pridėtinis triukšmo kiekis. Triukšmui sugeneruoti buvo paruošti trys algoritmai: normaliojo skirstinio, druskos ir pipirų metodas su dviem skirtingais sprendimais. Tyrimui atlikti buvo parašytos linijų radimo ir uždėjimo funcijos, kur linijų paieškai buvo pasirinkta „OpenCV“ bibliotekos „Hough“ transformacijos ir tikimybinės „Hough“ transformacijos algoritmai, įgyvendintas parametrų švelninimo žingsnis, atmetamos netinkamos linijos (įstrižos ir vertikalios), bei linijos esančios prie nuotraukos kraštų. Triukšmui sumažinti buvo naudojamas Gauso filtras. Darbe sudarytas testavimo modelis, padėjęs nustatyti tinkamiausią kraštų ir linijų radimo algoritmus. Šie algoritmai buvo naudoti tiesinimo modelio sukūrimui, kurio gauti rezultatai bandymų metu, keičiant jo pagrindinį parametrą MPN (maksimalų pasuktų nuotraukų skaičių), lyginami su nustatytomis teisingų kreipimo kampų (TKK) reikšmėmis. Sudarytas tiesinimo modelis taip pat buvo palygintas su trimis RotNet modeliais, kuriems apmokyti buvo parinkti trys skirtingi duomenų rinkiniai ir vienas modelis apmokytas su visais trimis duomenų rinkiniais. Šie RotNet modeliai taip pat naudoti testinių nuotraukų tiesinimui, gauti tiesinimo kampai buvo lyginti su TKK reiškmėmis.

Summary

Automatic algorithms for processing of images

Three edge detection algorithms are being analyzed in this master final paper work: „Sobel“, „Scharr“ and „Canny“. There is full analysis done in this paper about how these three algorithms are working with pictures that represent real life objects, moreover to that how algorithms are dealing with noisy images and what impact various levels of additional noise are making to detection of edges. There were three algorithms made to add noise: normal distribution (two levels of noise, when $\mu = 0$, o $\sigma = 20$ and $\mu = 0$, o $\sigma = 60$) and two solutions of salt and pepper method: one is generating noise using random numbers method and checking with defined possibility value (two options for possibility value were used: $prob = 0.005$ and $prob = 0.3$), the second solution is using thresholds that are being calculated by median number of gray image's pixels and σ where two values were selected for it: $\sigma = 0.005$ and $\sigma = 0.33$. Two functions were implemented to find and add lines on the original pictures for this research paper practical part: to find lines two „OpenCV“ library functions were in use - „Hough“ and Probabilistic Hough Transforms, also the parameters mitigation step was implemented, unnecessary lines like verticals or diagonals were rejected as well as lines that are going too close picture's frame. Gauss filter function from „OpenCV“ library was in use to reduce noise in the noisy pictures with parameters kernel size 7 and $\sigma = 60$, later it was noticed that for noisy images itself it is better to use Gauss filter with parameters when kernel size is 5 and $\sigma = 0$. Test model was made in this master paper work for helping out to find which edge detection algorithm is the best for further lines searching and identify how written functions for finding and adding lines on the image managed to do a job with pictures where various levels of noise were added. According these testing results, the best edge and lines detection algorithms were in use to create straightening pictures model. By changing his the most important parameter MPN - maximum rotated pictures number - to 1, 5, 10, 50 and 100, the results were gotten and rotation angles were compared with defined TKK - the correct rotation angles - values. Also in this pictures rotation model the best parameters for Gauss filter were chosen for each picture. This straightening pictures model was compared with how three RotNet models, that were trained using various data sets (pictures count in data sets were also various) and one additional which was trained with all pictures from three data sets, were able to rotate testing pictures according TKK values.

Įvadas

Paveikslėlių redagavimas yra viena iš kebliausių užduočių, ypač tuomet, kai reikia redaguoti šimtus nuotraukų vienu prisėdimu. Kuomet nuotraukų redagavimas yra hobis daugumos žmonių, kartais šis hobis pavirsta galvos skausmu, norint ištiesinti, pašviesinti ar kitaip redaguoti nuotrauką. Šiame darbe orentuojamasi į nuotraukos tiesinimo uždavinį. Dirbant daug su nuotraukomis, kartais keblu ištiesinti, ir rasti teisingą tiesinimo kampą patiems, taigi kyla klausimas – ar kompiuteris ir atitinkamas algoritmas gali ištiesinti nuotrauką už žmogų automatiškai?

Tiesinimo uždavinys susideda iš kelių komponentų: kraštų aptikimo paveikslėlyje, linijų radimo, kurios atitiktų horizontalias linijas nuotraukoje, paveikslėlio pakreipimo (ištiesinimo). Fotografijoje naudojamos horizontalios linijas tikriausiai daugiau, nei net yra suvokiama. Beveik kiekvienoje kraštovaizdžio nuotraukoje yra horizontali linija arba bent numanoma horizontali linija, ir tai – horizontas. Horizontas yra stabilus, patikimas ir nejudrus, todėl, kad jis yra horizontaliųjų linijų viršūnė, visos mažesnės horizontaliosios linijos taip pat įgyja tą prasmę [24]. Taigi darbo metu bus koncentruojamasi į pasvirusias horizontalias linijas, kuriomis remiantis, bus tiesinamas paveiksliukas.

Kraštų aptikimas apima įvairius matematinius metodus, turinčius tikslą nustatyti skaitmeninio vaizdo taškus, kuriuose vaizdo ryškumas smarkiai keičiasi arba, kitaip tariant, atsiranda vaizdo netolygumai. Taškai, kuriuose akivaizdžiai keičiasi vaizdo ryškumas, paprastai yra suskirstyti į lenktų linijų segmentų, vadinamų kraštais, rinkinį [35].

Gautas kraštų rinkinys po kraštų aptikimo operatoriaus pritaikymo su analizuojamu skaitmeniniu paveiksliuku, gali atvaizduoti paveikslėlyje esančių objektų ribas, paviršiaus žymėjimo ribas, taip pat kreives, atitinkančias paviršiaus netolygumus. Dėl to kraštų aptikimo algoritmo taikymas gali žymiai sumažinti apdorojamų duomenų kiekį, todėl gali būti filtruojama mažiau aktuali informacija, išlaikant svarbias vaizdo struktūrines savybes. Jei briaunų aptikimo žingsnis bus sėkmingas, tolesnė užduotis aiškinti originaliame paveikslėlyje pateiktą informacijos turinį tampa supaprastinta. Tačiau ne visada įmanoma gauti tokius idealius kraštus iš realaus gyvenimo vaizdų [21].

Pasikartojanti skaitmeninio vaizdo apdorojimo problema yra tiesių linijų aptikimas. Pirmas žingsnis ieškant tiesių linijų paveikslėliuose yra kraštų rinkinys, gautas po kraštų aptikimo operatoriaus pritaikymo. „Hough“ transformacija, kuri padeda kraštų taškus rastus paveikslėlyje sugrupuoti į objektų kandidatus, atliekant skirstymo procesą, yra plačiai naudojama objektų išskyrimui ir objektų aptikimui. Tačiau „Hough“ transformacijos metu atskiri vaizdo elementai, mūsų atveju galimos linijos, balsuoja už daugelį galimų parametrų įverčių. Dėl to susidaro tankus susumuotas masyvas ir kyla problemos nustatant parametrų reikšmes, kurios atitinka galimas tieses paveikslėlyje. Išspręsti linijų radimo problemas, į pagalbą bus pasitelktas dirbtinis intelektas. Naujas metodas naudoja konkurencinio neuroninio tinklo algoritmą atlikti tikimybines išvadas [16].

Darbo metu naudotos nuotraukos atvaizduoja realaus gyvenimo vaizdus. Tiesių linijų paieškos uždavinio pasunkinimui, darbo metu bus sugeneruotas skirtingo tipo triukšmas ir pridėtas prie originalių nuotraukų. Analizei ir tyrimui atlikti pasirinkti trys kraštų aptikimo algoritmai: „Sobel“, „Scharr“ ir „Canny“, o tiesių linijų paieškai: „Hough“ ir tikimybinė „Hough“ transformacija.

Pagrindinė **metrika** šiame darbe yra kampo reikšmė, skirta ištiesinti nuotraukas. Nuotraukų tiesinimo modelių pasiūlyti tiesinimo kampai bus lyginami su nustatyta kampų reikšmėmis **TTK**, nustatytoms kiekvienai teste dalyvavusiai nuotraukai (viso 50 nuotraukų) - ši reikšmė gauta, kai nuotraukos buvo rankiniu būdu pasuktos naudojant redegavimo įrankį, taigi **TTK** reikšmė reiškia, kokiu kampu nuotraukas reikia atstatyti, kitaip tariant - ištiesinti. Plačiau apie tai skyrelyje 4.

Šio darbo **tikslas** – ištiesinti nuotraukas naudojant dirbtinį neuroninį tinklą ir pasiūlytą tiesinimo metodą, kuriame panaudojami kraštų ir linijų aptikimo algoritmai.

Šiam magistro baigiamajam darbui išsikelti **uždaviniai**:

1. Palyginti ir pritaikyti kraštų aptikimo algoritmus ir rasti tiesias linijas originaliose ir užtriukšmintose nuotraukose.
2. Iš gautų rezultatų parinkti tinkamiausius kraštų aptikimo ir linijų radimo algoritmus, sudaryti nuotraukų tiesinimo modelį.
3. Palyginti gautus ištiesintų nuotraukų kampus su TTK parametro reikšme - nustatyta teisinga tiesinimo kampo reikšme kiekvienai teste dalyvavusiai nuotraukai.
4. Surinkti duomenų rinkinius skirtus RotNet neuroninio tinklo apmokymui.
5. Apmokyti RotNet tinklą su skirtingais surinktais duomenų rinkiniais ir palyginti ištiesintų nuotraukų kampus su TTK reikšmėmis.
6. Palyginti RotNet modelių gautus rezultatus su sukurto nuotraukų tiesinimo modelio gautais rezultatais.

Šis magistro baigiamasis darbas yra tęstinis: kraštų algoritmų ir linijų radimų algoritmų tyrimas buvo atliktas tiriamojo darbo metu, tie skyreliai yra pipildyti. Darbo struktūra: 1-as skyrius yra teorijos apžvalga, po jo seka kraštų aptikimo ir linijų radimo testavimo modelio aprašymas testų rezultatų aptarimas, 4-as skyrius skirtas apžvelgti, kokios nuotraukos tinkamos ir buvo parinktos tiesinimo uždaviniui atlikti, 5-as skyrius yra tiesinimo modelio aprašymas ir tyrimas, 6-as skyrius skirtas RotNet modelių tyrimo apžvalgai ir palyginimui su tiesinimo modeliu. 7-as skyrius skirtas programavimo dalies aprašymui.

Svarbiausi skyreliai: 7 programavimo dalies aprašymas, 2.2 užtriukšminimas, 2.4 linijų aptikimo algoritmai, 5 nuotraukos tiesinimo modelis, 6.3 RotNet tinklo tyrimas.

Panašių darbų apžvalga

Žemiau pateikta panašių darbų apžvalga. Šie darbai buvo motyvacija šiam magistro baigiamajam darbui įgyvendinti.

Ruba Anas, Hadeel A. Elhadi ir Elmustafa Sayed Ali parašytame straipsnyje „Kraštų aptikimo algoritmų daroma įtaka medicinos nuotraukų apdorojimui“ (angl. *Impact of Edge Detection Algorithms in Medical Image Processing*) [1], yra pristatoma galimi triukšmo variantai esantys skirtingo tipo medicinos nuotraukose, tokie kaip impulsyvus, druskos ir pipirų metodo sugeneruotas, Gauso pasiskirtymo, „nuodingasis“ (angl. *poisson*) ir taškelių (angl. *speckle*) triukšmai. Straipsnyje yra aprašoma, kaip kraštų aptikimo algoritmai (straipsnyje tiriami algoritmai: „Canny“, „Laplacian“, „LOG“ ir „Prewitt“) susidoroja su skirtingo Gauso pasiskirtymo ir druskos bei pipirų metodu užtriukšmintomis medicininėmis nuotraukomis. Buvo nustatyta, kad geriausias algoritmas kraštų atvaizdavimui po druskos bei pipirų triukšmo pašalinimo yra „Prewitt“ algoritmas, kai panaudotas triukšmo nuėmimo medianos filtrą. Be to, buvo įrodyta, kad geriausias kraštų aptikimo po Gauso triuškmo pašalinimo yra „Canny“ algoritmas, triukšmo nuėmimo medianos filtrą.

M. W. Spratling'o straipsnyje „Neuroninis „Hough“ transformacijos įgyvendinimas paaiškinimo būdu“ (angl. *A neural implementation of the Hough transform and the advantages of explaining away*) [34] yra aprašomi neuroninio tinklo algoritmai, jų įgyvendinimas ir veikimo palyginamas su „Hough“ transformacijos veikimu. Tyrimo metu buvo nustatyta, kad pasiūlytas metodas tiksliau identifikuoja tieses ir tų linijų parametrus nuotraukose, palyginti su standartiniu „Hough“ transformacijos balsavimo procesu.

Daniel Saez straipsnyje „Nuotraukų padėties taisymas naudojant konvoliucinius neuroninius tinklus“ (angl. *Correcting Image Orientation Using Convolutional Neural Networks*) [29] yra aprašomas RotNet modelio veikimas „Google Street“ vaizdų tiesinimui išspręsti. Modelio, apmokyto su „Google Street“ nuotraukomis paklaidos reikšmė tik 1-2 laipsniai.

1. Teorinė dalis

1.1. „Sobel“ operatorius

„Sobel“ kraštų aptikimo operatorius naudoja du 3×3 kernelius, kurie dauginami su originaliu paveikslėliu, kad apskaičiuotų išvestinių apytikslius duomenis - vieną horizontaliems pokyčiams, kitą vertikaliems. Jei apibrėžtume *I'* kaip originalų vaizdą (ar sulietą, gautą po triukšmo šalinimo), o G_x ir G_y yra du vaizdai, kurių kiekviename taške yra atitinkamai vertikalios ir horizontalios išvestinių aproksimacijos, gautos sandaugos G_x ir G_y tarp paveikslėlio *I'* ir „Sobel“ filtro abiejų krypčių (horizontalios ir vertikalios) kerneliais funkcijos išvedama taip 1.1:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I' \quad \text{ir} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I'. \tag{1.1}$$

„Sobel“ kraštų aptikimo kerneliai yra suprojektuoti maksimaliai reaguoti į kraštus, einančius vertikaliai ir horizontaliai pikselių tinklelio atžvilgiu, po vieną kernelį kiekvienai iš dviejų statmenų koordinačių (*x* ir *y*). „Sobel“ algoritmo metu gautos sandaugos G_x ir G_y atskirai ir apjungtos palyginamos su originaliu paveiksliuku parodytos pav. 1.

Gautos sandaugos G_x ir G_y gali būti apjungtos, padedant nustatyti absoliučius nuolydžio dydžius kiekviename taške ir to gradiento orientaciją. Taigi, gradiento dydžio $|G|$ formulė 1.2:

$$|G| = \sqrt{(G_x)^2 + (G_y)^2}. \tag{1.2}$$

Naudojant tuos pačius duomenis, galima apskaičiuoti gradiento kryptį su formule 1.3:

$$\theta = arctan(\frac{G_y}{G_x}). \tag{1.3}$$

Šiuo atveju θ, kurios reikšmė 0 reiškia, kad maksimalaus kontrasto nuo juodos iki baltos kryptis vaizde rodoma iš kairės į dešinę, o kiti kampai matuojami prieš laikrodžio rodyklę. Kaip pavyzdys triukšmas gali turėti didelę kampo reikšmę, kuri paprastai yra nepageidaujama [26]. Naudojant gradiento kampo informaciją, vaizdo apdorojimo programos gali padėti pašalinti vaizdo triukšmą, daugiau apie tai poskyryje 1.3.3.

1 pav. G_x ir G_y gradientų palyginimas su originalia nuotrauka (nespalvota) ir po pilno „Sobel“ algoritmo pritaikymo. Nuotrauka nr. 1.

1.2. „Scharr“ operatorius

„Scharr“ operatorius yra alternatyva „Sobel“ operatoriui, nors veikimo principas toks pats. Šis operatorius bando pasiekti tobulą sukimosi simetriją [17]. Skirtumas kerneliuose, jeigu G_x ir G_y yra du vaizdai, kurių kiekviename taške yra atitinkamai vertikalios ir horizontalios išvestinių aproksimacijos, gautos sandaugos G_x ir G_y tarp paveikslėlio *I'* ir „Scharr“ filtro abiejų krypčių (horizontalios ir vertikalios) kerneliais funkcijos išvedama taip 1.4:

$$G_x = \begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix} * I' \quad \text{ir} \quad G_y = \begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} * I'. \tag{1.4}$$

„Scharr“ algoritmo metu gautos sandaugos G_x ir G_y atskirai ir apjungtos palyginamos su ori-

ginaliu paveiksliuku parodytos pav. 2. „Scharr“ operatorius suteikia intensyvumo rastų kraštų pikseliams.

2 pav. G_x ir G_y gradientų palyginimas su originalia nuotrauka (nespalvota) ir po pilno „Scharr“ algoritmo pritaikymo. Nuotrauka nr. 1.

1.3. Canny algoritmas

„Canny“ kraštų aptikimo detektorius yra iš kelių žingsnių sudarytas algoritmas, aptinkantis įvairius kraštus esančius nuotraukose. John'as F. Canny sukūrė „Canny“ algoritmą 1986-ias [5].

„Canny“ algoritmą galima suskirstyti į penkis skirtingus žingsnius [31]:

1. Pašalinti triukšmą esantį nuotraukoje naudojant Gauso filtrą.
2. Rasti nuotraukos intensyvumo gradientus (angl. *gradients*).
3. Pritaikyti ne-maksimumo (angl. *non-maximum*) slopinimą.

4. Pritaikyti dvigubą ribą (angl. *threshold*), padėsiančią nustatyti galimus kraštus.

5. Panaudoti histerezę: užbaigti kraštų aptikimą, nuslopinant visus kitus kraštus, kurie yra silpni ir neprijungti prie stiprių briaunų.

1.3.1. Triukšmo šalinimas

Kadangi kraštų aptikimas priklauso nuo triukšmo esančio vaizde, pirmiausia reikia jį pašalinti. „Canny“ algoritmas naudoja Gauso filtrą triukšmui šalinti. Pritaikius Gauso filtrą, nuotrauka atrodo susiliejusi. Nuotraukos suliejimas gaunamas, pritaikant Gauso funkciją (kuri taip pat reiškia normalų pasiskirstymą statistikoje), skaičiavimo transformacija taikoma kiekvienam paveikslėlio taškui (pikseliui).

Gauso funkcija vienoje dimensijoje formulė yra 1.5:

$$f(x) = \alpha * exp^{-\frac{x-b}{2\sigma^2}}, \tag{1.5}$$

kur α, b ir σ yra konstantos. Gauso funkcijos gautas grafikas yra „varpo formos“, tokiam grafike α (pilna formulė 1.6) atvaizduoja grafiko kreivės smailės aukštį, b centrinio taško pozicija ir σ standartinis nuokrypis - dydis nusakantis atsitiktinio dydžio įgyjamų reikšmių sklaidą apie vidurkį, kitaip tariant parametras, kuris atsakingas už „varpo“ formos pločio kontroliavimą.

$$\alpha = \frac{1}{2\pi\sigma^2}. \tag{1.6}$$

Gauso filtro funkcija, skirta rasti transformaciją kiekvieno paveikslėlio *I(i,j)* taško yra 1.7:

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x+y^2}{2\sigma^2}), \tag{1.7}$$

kur *x* ir *y* yra horizontalios ir Gauso kernelio (*angl.* Gaussian kernel) *H(x,y)* vertikalios dimensijos, kurios konvuliuoja (*angl.* convolute) su paveikslėliu *I(i,j)* ir σ - Gauso pasiskirtymo standartiniu nuokrypiu. Konvoliucija, kurios metu gaunamas naujas paveikslėlis *I'(i',j')* tarp paveikslėlio *I(i,j)* ir Gauso kernelio *H(x,y)* išreiškiama formule 1.8 [20]:

$$I'(i',j') = \sum (x,y) \in_{R_H} I(i'-x, j'-y) \times H(x,y). \tag{1.8}$$

Gauso filtro kernelio, kurio dydis $(2k+1) \times (2k+1)$ funkcija 1.9:

$$H_{ij} = \frac{1}{2\pi\sigma^2} exp(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}); 1 \leq i,j \leq (2k+1). \tag{1.9}$$

Kernelio dydis priklauso nuo kaip labai paveikslėlis turi būti sulietas - kuo didesnis kernelio dydis, tuo didesnis suliejimo efektas, taip pat ir didinat σ reikšmę [31]. Pavyzdys parodytas lentelėje 3, kur labiausiai susiliejusi nuotrauka gauto pritaikius Gauso filtrą, kuomet kernelio dyds $K = 7 \times 7$, o $\sigma = 60$.

Originalas

$K = 7 \times 7, \sigma = 60$

$K = 3 \times 3, \sigma = 0$

$K = 3 \times 3, \sigma = 30$

$K = 5 \times 5, \sigma = 0$

$K = 5 \times 5, \sigma = 30$

3 pav. Nuotrauka po Gauso filtro panaudojimo. Nuotrauka nr. 1.

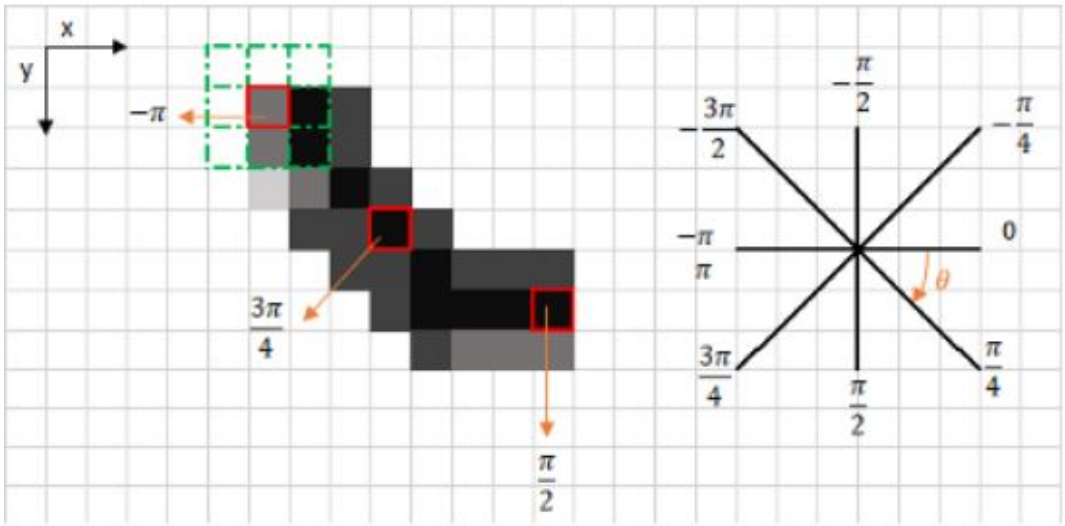
1.3.2. Gradientų skaičiavimas

Gradientas – diferencialinis operatorius, skaliarinį lauką atvaizduojantis į vektorinį lauką taip, kad kiekvienas vektorinio lauko vektorius būtų nukreiptas skaliarinio lauko reikšmių didėjimo kryptimi, o jo modulis būtų lygus kryptinei išvestinei šiame taške [23]. Gradiento skaičiavimo nuotraukoje žingsnis nustato krašto ryškumą ir kryptį, apskaičiavus vaizdo gradientą, skaičiavimams naudojamas krašto aptikimo operatorius, „Canny“ algoritmo atveju naudojamas „Sobel“ kraštų aptikimo operatorius aprašytas skyriuje 1.1 [31].

1.3.3. Ne-maksimumo (angl. *non-maximum*) slopinimas

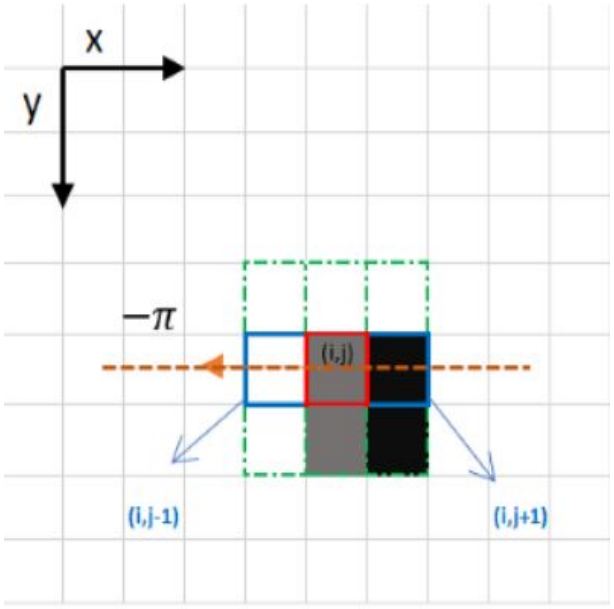
Ne-maksimumo slopinimas yra kraštų retinimo technika. Ne-maksimumo slopinimas taikomas norint rasti „ryškiausią“ kraštą. Atlikus gradiento skaičiavimą (žingsnis 1.3.2), kraštas, paimtas iš gradiento vertės, vis dar yra gana neryškus, turėtų būti tik vienas tikslus krašto pažymėjimas. Taigi, ne-maksimumos slopinimas gali padėti slopinti visas nuolydžio vertes (jas nustatant 0), išskyrus maksimumo vertes, kurios nurodo vietas, kuriose yra ryškiausias krašto pokytis.

Toliau nagrinėjamas pavyzdys: viršutinis kairiojo kampo raudonas langelis pavaizduotas pav. 4, rodo apdorojamo gradiento intensyvumo matricos pikselį. Atitinkamą briaunos kryptį rodo oranžinė rodyklė, kurios kryptis radianais yra $-\pi$ (+/– 180 laipsnių).



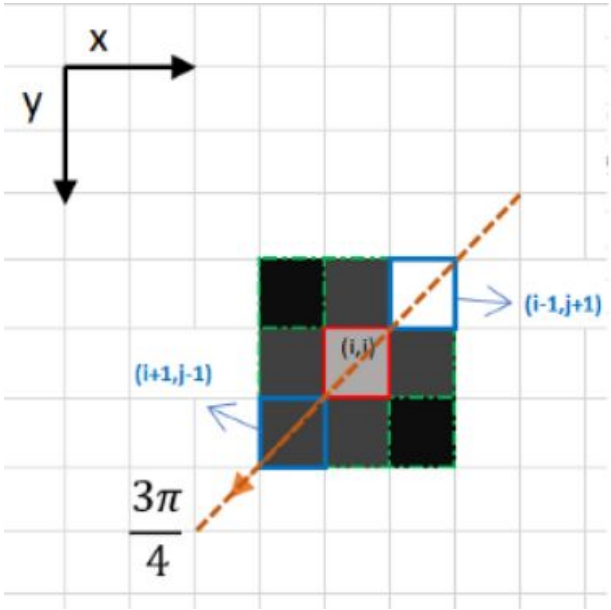
4 pav. Pikselių išsidėstymo ir krypčių pavyzdys [31]

Ne-maksimumo slopinimo algoritmo veikimo pavyzdys pav. 5, kur krašto kryptis yra oranžinė punktyrinė linija (horizontali linija einanti iš kairės į dešinę). Algoritmo tikslas yra patikrinti, ar tos pačios krypties taškai yra daugiau ar mažiau intensyvūs nei apdorojamas taškas. Aukščiau pateiktame pavyzdyje apdorojamas pikselis (i, j), o ta pačia kryptimi esantys pikseliai yra paryškinti mėlynai - $(i, j-1)$ ir $(i, j+1)$. Jei vienas iš šių dviejų mėlynų pikselių yra intensyvesnis nei apdorojamasis, tada išlaikomas tik intensyvesnis. Pikselis $(i, j-1)$ atrodo intensyvesnis, nes yra baltos spalvos, kurios vertė 255. Taigi dabartinio taško (i, j) intensyvumo reikšmė prilyginama 0. Jei tarp krašto pikselių nėra intensyvesnės reikšmės pikselio, tada dabartinio apdorojamo taško vertė yra išlaikoma.



5 pav. Pikselių 3×3 ne-maksimumo slopinimo pavyzdys (1) [31]

Kitame pavyzdyje pav. 6 krašto kampas yra $\frac{3\pi}{4}$ ir intensyviausias pikselis yra $(i-1, j+1)$ (baltos spalvos, 255 vertės).



6 pav. Pikselių 3×3 ne-maksimumo slopinimo pavyzdys (2) [31]

Apibendrinant ne-maksimumo slopinimo žingsnį, galima pasakyti, jog kiekvienas pikselis turi 2 pagrindinius kriterijus: briaunų kryptis radianais ir pikselių intensyvumą, kurių vertė tarp 0–255, jeigu nuotrauka yra juodai - balta. Spalvotos nuotraukos turės daugiau triukšmo, todėl siekiant tiksliausių rezultatų atliekant „Canny“ algoritmo žingsnius, verta nuotrauką paversti į nespalvotą prieš pradedant.

Žinant du pikselių kriterijus, ne-maksimumo slopinimo algoritmo žingsniai yra tokie:

1. Apibrėžti matricą inicijuotą tik 0 reikšmėmis, matricos dydis toks pats kaip gautų gradientų matricos.

2. Gauti krašto kryptį pagal kampo vertę iš kampų matricos.

3. Patikrinti, ar ta pačia kryptimi esantis pikselis yra intensyvesnis nei šiuo metu apdorojamas pikselis.

4. Grąžinti gautą paveikslėlį po ne-maksimumo nuslopinimo.

Gautas rezultatas yra tas pats vaizdas su plonesnėmis briaunomis. Tačiau vis tiek galime pastebėti kai kuriuos briaunų ryškumo skirtumus: kai kurie taškai atrodo ryškesni nei kiti, ir šis trūkumas yra pašalinimas likusiais dviem žingsniais [31].

1.3.4. Dviguba riba (angl. *threshold*)

Atliekant dvigumos ribos žingsnį, siekiama nustatyti trijų tipų vaizdo elementus - stiprius, silpnus ir nereikšmingus:

- Stiprūs taškai yra tokie, kurių intensyvumas yra toks didelis, jog galima teigti, kad jie prisideda prie galutinio krašto vaizdo.
- Silpni taškai yra tokie, kurių ryškumo vertė nėra pakankama, kad būtų laikoma stipria, bet dar nėra pakankamai maža, kad būtų laikoma nereikšminga krašto aptikimui.
- Kiti taškai yra laikomi nereikšmingais krašto radimui (nepriklauso kraštui).

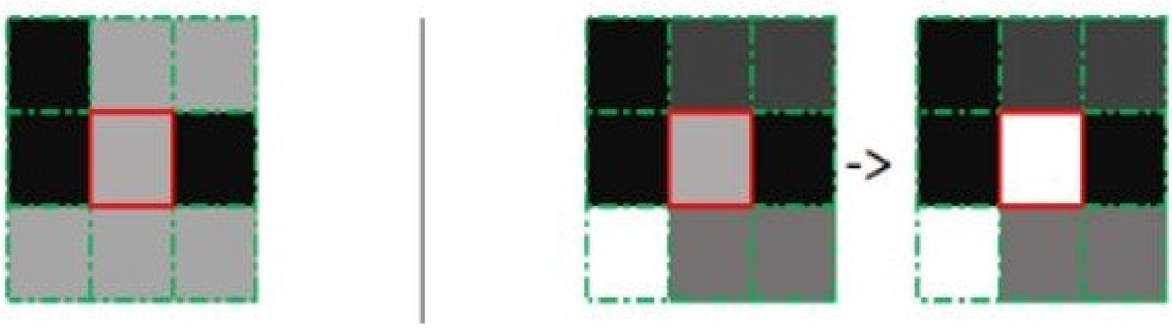
Taigi, dvigubos ribos žingsnio tikslas yra:

- Aukštos ribos vertė padeda identifikuoti ryškiausius vaizdo taškus (intensyvumas didesnis nei nustatyta aukštos ribos reikšmė).
- Žemos ribos vertė nustato nereikšmingus pikselius (intensyvumo reikšmė mažesnė už nustatytą žemos ribos reikšmę).
- Pikseliai, kurių intensyvumo vertė patenka tarp žemos ir aukštos ribos reikšmių, pažymimi kaip silpni taškai.

Šio žingsnio rezultatas yra paveikslėlis turintis tik dvi pikselių intensyvumą nustatančias reikšmes: stiprią ir silpną [31].

1.3.5. Histerezės panaudojimas

Histerezės mechanizmas (paskutinis žingsnis) padės atpažinti tuos taškus, kurie gali būti laikomi stipriais, ir tuos, kurie laikomi nereikšmingais. Remiantis dvigubos ribos gautais rezultatais, histerezę sudaro silpnų pikselių pavertimas stipriais, jei ir tik tada, jei bent vienas iš aplink apdorojamą vaizdo elementų yra stiprus, pavyzdys parodytas 7 pav.



7 pav. Kairėje pusėje 3×3 matricos pikselių pavyzdys, kur aplinkui apdorojamą pikselį nėra stipraus intensyvumo pikselio, todėl jo intensyvumas nėra keičiamas. Dešinėje pusėje esančioje pikselių matricijoje, apdorojamas pikselis paverčiamas stipriu, nes aplinkui aptinkamas stiprus pikselis. [31]

1.4. „Canny“ algoritmas iš „OpenCV“ bibliotekos

„Canny“ algoritmas naudotas „OpenCV“ bibliotekos. Šios bibliotekos algoritmo implementacija nevykdo Gauso filtro funkcijos. „Canny“ algoritmas priima tokius parametrus: apdorojamą paveiksliuką, apatinę ribos reikšmę ir viršutinę ribos reikšmę, kurios naudojamos dvigubos ribos žingsnyje aprašytame poskyryje 1.3.4 [9].

Dvigubos ribos optimalių reikšmių parinkimas pagrįstas tvirtos (angl. robust) statistikos įverčiu: paskaičiuojama nuotraukos medianos reikšmė - vidurinė variacinės eilutės reikšmė [37], tiksliau nei vidurkis padeda aptikti, kurios reikšmės yra anomalijos (toliausiai esančios nuo paplitusios reikšmės), tuo tarpu vidurkis gali iškraipyti pasiskirstymo rezultatą. Viena didelė anomali reikšmė variacijos eilutėje gali paslinkti vidurkio reikšmę toliau nuo labiausiai paplitusių reikšmių [14].

Kitas svarbus parametras skaičiuojant apatinė ir viršutinę ribą yra σ - standartinis nuokrypis – dydis nusakantis atsitiktinio dydžio įgyjamų reikšmių sklaidą apie vidurkį, mūsų ateveju - mediana [19]. Remiantis Adrian'o Rosebrock'io atliktu tyrimu [27], optimaliausi kraštų aptikimo rezultatai gauti parinkus $\sigma = 33\%$, ši reikšmė bus naudota ir toliau aprašytame tyrime. Taigi, apatinės (formulėje žymima *AR*) ir virštunės (formulėje žymima *VR*) ribos apskaičiavimas išreiškiamas formule 1.10, atsižvelgiant, jog nuotrauka nespalvota (pikselių reikšmės nuo 0 iki 255):

$$AR = \max[0, (1 - \sigma) \times Mediana] \quad \text{ir} \quad VR = \min[255, (1 + \sigma) \times Mediana]. \tag{1.10}$$

Po visų žingsnių gautas nespalvoto originalaus paveiksliuko kraštų atvaizdas pademonstruotas pav. 8.

8 pav. „Canny“ algoritmo pritaikymas. Nuotrauka Nr.1.

1.5. „Hough“ transformacija

Kraštų aptikimo algoritmas tik pirmas žingsnis norint rasti tiesiausią liniją. Atliekant automatinę skaitmeninių vaizdų analizę, dažnai iškyla problema aptikti paprastas formas, tokias kaip tiesios linijos, apskritimai ar elipsės. Tačiau dėl vaizdo duomenų ar kraštų aptikimo algoritmo trūkumų gali atsirasti trūkstamų taškų, kurie reikalingi išgauti norimas kreives, taip pat erdvinių nuokrypių tarp idealios linijos, apskritimo ar elipsės ir triukšmingų krašto taškų, kurie gaunami po briaunų operatoriaus pritaikymo. Dėl šių priežasčių nėra trivialu suskirstyti gautus taškus, jog šie sudarytų linijos, apskritimo ar elipsės vaizdą. „Hough“ transformacijos tikslas yra išspręsti šią problemą - kraštų taškai rasti paveikslėlyje sugrupuojami į objektų kandidatus, atliekant skirstymo procesą. Toliau bus tiriama linijų objektų radimas, naudojantis „Hough“ transformacijos pagalba [32].

Parametrų atvazdavimas, naudojamas apibūdinti linijas esančias paveikslo plokštumoje, sudaro parametrų erdvę. Tiesė $y = mx + b$ gali būti pavaizduota kaip taškas (b, m) parametrų erdvėje, tačiau vertikalios linijos kelia problemų. Deja, tiek nuolydžio parametras m tiek fiksuotas dydis nusakantis tiesios linijos kitimą b nėra apriboti, o tai apsunkina tiesių paieškos procesą. Taigi dėl skaičiavimo priežasčių O. Duda ir E. Hart'as [13] pasiūlė naudoti „Heseno“ normaliąją formą, kuri parodyta funkcijoje 1.11:

$$r = x \cos\theta + y \sin\theta, \tag{1.11}$$

kur r yra atstumas nuo pradžios taško iki artimiausio taško tiesioje linijoje, o θ yra kampas tarp x ašies ir linijos, jungiančios pradžios tašką su tuo artimiausiu tašku linijoje.

1.5.1. Realizacija

Su kiekviena vaizdo eilute galima susieti porą (r, θ). Porų (r, θ) plokštuma kartais vadinama dvimate „Hough“ erdve skirta tiesių linijų rinkiniui. Taigi algoritmas pirmiausia inicijuoją dvimatę

matricą šioms porų reikšmėms laikyti, pradinės reikšmės nustatomos 0. Matricos eilutės yra r reikšmės, o stulpeliai yra θ reikšmės. θ masyvo dydis priklauso nuo reikalingo tikslumo. Tarkime, kad poreikis yra, jog kampų tikslumas būtų 1 laipsnis, tuomet reikia 180 stulpelių. Didžiausias įmanomas atstumas paveikslėlyje yra jo įstrižainės vaizdas. Taigi, atsižvelgiant į vieno laipsnio tikslumą, eilučių skaičius gali būti lygus paveikslėlio įstrižainės ilgiui.

Imamas kiekvienas gauto paveikslėlio po kraštų aptikimo algoritmo pritaikymo pikselis (x, y) ir atsižvelgiama į visas linijas, kertančias per tą tašką pasirinktu kampų rinkiniu (jeigu tikslumas vienas laipsnis, tai nuo 1 iki 180 laipsnių imtinai). Apskaičiuojams atstumas r iki galimos briaunos (rastos briaunos po kraštų radimo algoritmo pritaikymo, t.y. šviesaus taško) su kiekvieno kampo reikšme θ iš pasirinkto kampų rinkinio. Jeigu r negautas, reiškia nebuvo nė vieno šviesaus taško linijoje, kitu atveju, „Hough“ algoritmo sumavimo įrenginys (angl. *accumulator*) padidina per vienetą masyvo poros reikšmę (r, θ). Kitais žodžiais tariant, vyksta balsavimas už (r, θ) vertes. Po įvykdyto visų linijų įvertinimo per visus taškus, didelės vertės pora (r, θ) „Hough“ erdvėje (matrica) greičiausiai atitiks taškų liniją.

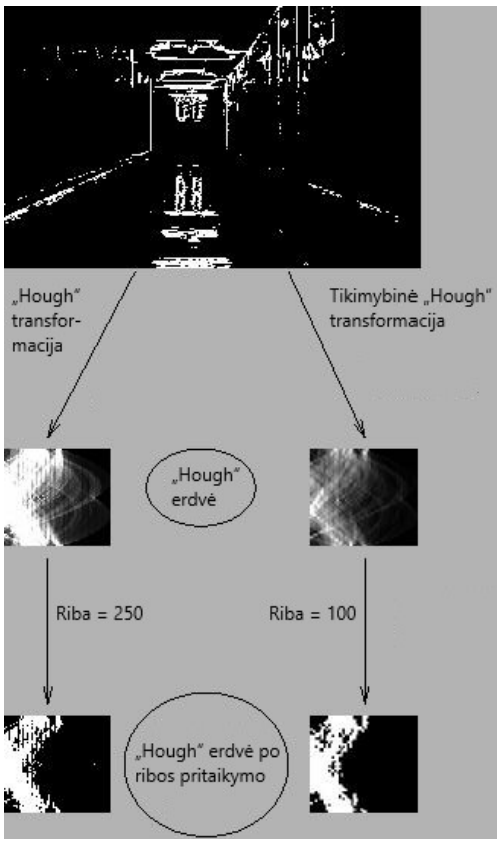
Gautos matricos langelių reikšmės rodo kreivių skaičių per bet kurį paveikslėlio tašką. Didesnės langelių vertės tampa ryškesnės [10].

1.5.2. „Hough“ transformacijos algoritmas iš „OpenCV“ bibliotekos

Viskas, kas paaiškinta praeitame poskyryje, yra įtraukta į „OpenCV“ bibliotekos funkciją - *cv2.HoughLines()*. Funkcija papraščiausiai grąžina dvimatį masyvą su užpildytomis (r, θ) vertėmis, kurios surinko daugiausiai balsų sprendžiant, kad buvo aptikta linija. „OpenCV“ bibliotekos funkcija *cv2.HoughLines()* priima tokius parametrus: pirmasis parametras yra rasti kraštai dvejetainiame formate, antrasis ir trečiasis parametrai nurodo r ir θ tikslumus, pagal numatymą, tyrime naudosime $r = 1$ ir $\theta = np.pi/180$. Ketvirtasis parametras nurodo ribą - minimali balsavimo vertės riba, kad atitinkamas taškas būtų laikomas linija. Balsų skaičius priklauso nuo pikselių skaičiaus eilutėje, taigi ši ribos vertė nurodo ir mažiausią linijos ilgį, kurį norima aptikti.

1.5.3. Tikimybinės „Hough“ transformacijos algoritmas iš „OpenCV“ bibliotekos

Atliekant „Hough“ transformaciją yra pastebėta, kad net ir linijai su dviem argumentais užtrunka daug skaičiuoti. Tikimybinė (angl. *probabilistic*) „Hough“ transformacija - tai įprasto „Hough“ transformacijos algoritmo optimizavimas. Tikimybinė „Hough“ transformacijos metu neatsižvelgiama į visus taškus, vietoj to imamas tik atsitiktinis taškų pogrupis ir to pakanka linijai aptikti, toks parinkimas implemetuotas pasitelkiant patikimu linijų aptikimu (angl. *Robust Detection*) aprašytu detaliau J. Matas, C. Galambosy ir J. Kittler'io straipsnyje *„Progressive Probabilistic Hough Transform“* [22]. Jeigu linijų neaptinkama dėl per didelio triukšmo ar nutriukšminimo, riba bus mažinama. „Hough“ transformacijos ir tikimybinės „hough“ transformacijos palyginimas pavaizduotas pav. 9. Paveikslėlis atvaizduoja, kad bendras gautų linijų rezultatas panašus [2].



9 pav. „Hough“ transformacijos ir tikimybinės „hough“ transformacijos palyginimas [2].

„OpenCV“ bibliotekos funkcija *cv2.HoughLinesP()* turi du papildomus parametrus;

- *“minLineLength“* - minimalus linijos ilgis. Trumpesni už šio parametro vertę linijų segmentai yra atmetami.
- *“maxLineGap“* - didžiausias leidžiamas tarpas tarp linijų segmentų, kad būtų galima traktuoti, jog jie sudaro vientisą liniją.

Ši funkcija iškart tiesiogiai grąžina linijos galinius taškus, tad nereikia papildomo skaičiavimo prieš atvaizduojant linijas ant originalaus paveiksliuko (prieš tai aprašyta funkcija grąžina parametrus ir reikia surasti visus taškus) [10].

1.6. Neuroniniai tinklai

Dirbtinis intelektas liudija, kaip akivaizdžiai auga atotrūkis tarp žmogaus ir kompiuterio galimybių. Tyrėjai ir entuziastai dirba su daugeliu šios srities aspektų, kurių vienas iš daugelio yra kompiuterinio vaizdo sritis. Šios srities tikslas yra suteikti kompiuteriams galimybę pažvelgti į pasaulį taip, kaip jį mato žmonės, suvokti jį panašiai ir netgi panaudoti žinias daugybei užduočių, tokių kaip vaizdų ir įrašų atpažinimas, vaizdo analizė ir klasifikavimas, žiniasklaidos (angl. *media*) atkūrimas, rekomendacijų sistemos, natūralios kalbos apdorojimas ir kt. Kompiuterinio matymo su giliu mokymusi pažanga buvo sukonstruota ir tobulinta laikui bėgant, visų pirma per vieną konkretų algoritmą, dirbtinio neuroninio tinklo rūšį, dažniausiai taikomą kompiuterinio vaizdo srityje, t.y., vaizdų atpažinimui, objektų radimui ir identifikavimui nuotraukose ir panašiai - konvoliucinį neuroninį tinklą [30][36].

Neuroniniai tinklai yra organizuoti į tarpusavyje sujungtus dirbtinių neuronų sluoksnius. Paprasčiau tariant, kiekvienas sluoksnis paima ankstesnio sluoksnio išvestį, taiko daugybę transformacijų ir išsiunčia savo išvestį į kitą sluoksnį. Pirmojo sluoksnio įvestis yra susieta su pirminiais duomenimis, kuriuos norime apdoroti (vaizdai, tekstas ir kt.), O paskutinio sluoksnio išvestis yra tai, ką norime nuspėti.

Kiekviename sluoksnyje vykstančių transformacijų tikslas yra apdoroti ypatybes. Mašininio mokymosi metu ypatybės yra atributai, kurie reprezentuoja vazdinius duomenis. Pavyzdžiui, jei lyginame du vaizdus, lengviau palyginti ypatybes, kurios vaizduoja tam tikrą struktūrą keliuose vaizdinio taškuose, nei lyginti pikselius po vieną.

Tradiciniuose mašininio mokymo algoritmuose šias ypatybes žmonėms reikėjo kruopščiai sukurti, o neuroninis tinklas mokosi pats apskaičiuoti optimaliausias užduoties ypatybes. Kiekvienas neuroninio tinklo sluoksnis sukuria ankstesniame sluoksnyje apskaičiuotas ypatybes, tam kad išmoktų aukštesnio lygio ypatybių. Pavyzdžiui, apdorodamas paveikslėlį su mašina neuroninio tinklo pirmasis sluoksnis gali apskaičiuoti žemo lygio yaptybes, tokias kaip kraštai, o paskutinis sluoksnis gali apskaičiuoti aukšto lygio funkcijas, pvz., ratų egzistavimą paveikslėlyje. Iš esmės, neuronų tinklai, turintys daugiau sluoksnių, gali išmokti aukštesnio lygio ypatybių, turėdami omenyje, kad jie yra mokomi turėdami daugiau duomenų, todėl sakoma, kad jie turi geresnius mokymosi gebėjimus. Daug sluoksnių turintys neuroniniai tinklai vadinami gilaus mokymo neuroniniais tinklais. Tai yra priežastis, kodėl tokio tipo mašininio mokymosi algoritmai paprastai vadinami giluminiu mokymusi (angl. *deep learning*).

Kiekviena neuroninio tinklo sąsaja turi atitinkamą skaitinį svorį. Šie svoriai yra neuroninio tinklo vidinė būsena. Jie yra atsakingi už skirtingas savybes, kurios apskaičiuojamos kiekviename sluoksnyje. Norint, kad neuroninis tinklas (ar bet kuris mašininio mokymosi algoritmas) veiktų, jį reikia apmokyti. Apmokymo proceso pradžioje svoriai atsitiktinai inicijuojami, todėl tinklas išveda atsitiktines prognozes. Apmokymų metu tinklas aprūpinamas dideliu kiekiu sužymėtų duomenų ir naudoja prognozavimo klaidą (angl. *error*) (klaidą tarp numatomos išvesties ir tikrosios išvesties), tam jog atnaujintų savo svorio skaitinę vertę. Kai tinklas neteisingai prognozuoja, prognozavimo klaida bus didelė. Dėl to svoriai bus atnaujinami proporcingai klaidos vertei, kad kitą kartą, kai tinklui bus apmokomas su tuo pačiu duomenų pavyzdžiu, prognozavimo paklaida bus mažesnė. Kitaip tariant, mokymų eigoje svoriai atnaujinami, kad būtų sukurtos geresnės savybės, padedančios tinklui geriau nuspėti. Po daug apmokymo kartojimų prognozavimo paklaida bus tokia maža, kad svoriai nebus reikšmingai atnaujinti. Tada galima tegiti, jog mokymo procesas baigtas. Funkcija, kuri apskaičiuoja numatymo paklaidą, paprastai vadinama nuostolių funkcija, o algoritmas, naudojamas apskaičiuojant kiekvieno svorio atnaujinimą apmokymo metu, vadinamas atgaliniu skleidimu (angl. *backpropagation*). Kai tinklas bus apmokytas, svoriai bus fiksuoti ir tinklas gali būti naudojamas kitiems duomenims nuspėti.

Neuroniniai tinklai įprastai apmokomi duomenų partijomis (angl. *batch*). Kiekvieną kartą, kai partija paduodama į tinklą, prognozavimo klaida yra vidutinė visai partijai, naudojant nuostolių funkciją, ir tinklas atnaujina savo skaitinį svorį, remdamasis gauta klaida. Jeigu tinklas mokomas vienu metu naudojant tik vieną pavyzdį, svorio atnaujinimai bus labai netikslūs, nes tinklas juos optimizuos, remdamasis atskirais pavyzdžiais. Kitu atveju, jei tinklui bus paduodamas visas apmokymui skirtas duomenų rinkinys, svorio atnaujinimai bus labai tikslūs, nes jie bus pagrįsti vidutine visų apmokymų imčių prognozavimo paklaida. Tačiau šiuo atveju tinklo apmokymas vyks lėtai, nes jis gali atnaujinti savo svorį tik tada, kai bus paduotas visas apmokymui skirtas rinkinys. Partijų naudojimas apmokymams yra geras kompromisas: išvengiama anomalaus atnaujinimo, kai naudojami atskiri duomenų vienetai, o apmokymo procesas yra greitesnis nei tuo atveju, jei būtų

naudojamas visas duomenų rinkinys, nes svoriai atnaujinami dažniau. Dažniausiai naudojamos vienos partijos dydžio reikšmės yra 64, 128, 256 arba 512 [29].

1.7. Konvoliucinis neuroninis tinklas

Konvolucinis neuroninis tinklas (toliau CNN) yra gilaus mokymosi algoritmas, kuris priima įvesties vaizdą, suskaičiuoja ir priskiria svorio skaitines reikšmes (mokomieji svoriai ir poslinkiai) įvairiems vaizdo aspektams ar objektams ir geba atskirti ieškomus objektus nuo kitų. Įvestis, tarkime spalvotas RGB paveikslėlis išskiria tris spalvų plokštumas - raudoną, žalią ir mėlyną. Galima įsivaizduoti, kaip skaičiavimai taptų intensyvūs, kai vaizdai turėtų matmenis, pavyzdžiui, 8K (7680×4320). CNN vaidmuo yra sumažinti vaizdus į lengviau apdorojamą formą, neprarandant jo ypatybių, kurios yra labai svarbios norint gauti tinkamą prognozę. Tai svarbu, kai norima sukurti architektūrą, kuri ne tik gerai mokosi ypatybių, bet ir yra pritaikoma dideliems duomenų rinkiniams [30].

CNN sudaro dvi pagrindines dalys, vadinamos sluoksniais [3]:

- Konvoliucinis ir telkimo sluoksniai: suskaido vaizdą į ypatybes ir jas analizuoja.
- Visiškai sujungtas sluoksnis: ima išvestį iš konvoliucinio ir sujungimo sluoksnių ir numato geriausią etiketę (angl. *label*) vaizdui apibūdinti.

Konvoliucinis sluoksnis ir telkimo sluoksniu kartu sudaro i-ąjį konvoliucinio neuroninio tinklo sluoksnį. Atsižvelgiant į vaizdų sudėtingumą, tokių sluoksnių skaičius gali būti padidintas, kad būtų galima geriau užfiksuoti žemo lygio detales, tačiau tai daro įtaką skaičiavimo trukmei [30].

1.7.1. Konvoliucinis sluoksnis

Konvoliucinis sluoksnis atlieka daugiausiai skaičiavimų konvoliuciniame neuroniniame tinkle. Iš esmės konvoliuciniai sluoksniai veikia panašiai kaip visiškai sujungti sluoksniai. Skirtumas tas, kad neuronai yra maži branduoliai, prijungti tik prie nedidelės įvesties dalies [29].

Konvoliucinio sluoksnio (toliau KS) tikslas yra iš įvesties vaizdo išgauti aukšto lygio funkcijas, pavyzdžiui, tokias kaip kraštai. CNN neturi apsiriboti tik vienu konvoliuciniu sluoksniu. Paprastai pirmasis KS yra atsakingas už žemo lygio savybių - kraštai, spalvos, gradiento orientacija ir kt. - fiksavimą. Pridėjus sluoksnių, architektūra taip pat prisitaiko prie aukšto lygio funkcijų, suteikdama mums tinklą, sugebantį analizuoti visapusiškai vaizdus duomenų rinkinyje [30].

Konvoliucinio tinklo įvestį - paveikslėlį, kompiuteris supranta kaip pikselių masyvą, priklausantį nuo paveikslėlio resoliucijos. Remiantis rezoliucija, matrica bus sudaryta is h - aukščio, w - pločio ir d - dimensijos. Pavyzdžiui, paveikslėlis kurios dydis $6 \times 6 \times 3$ bus RGB tipo (3 asocijuojasi su trimis dimensijomis ateinančiom iš spalvoto spektro RGB), o $6 \times 6 \times 1$ yra nespalvotos nuotraukos masyvų matrica.

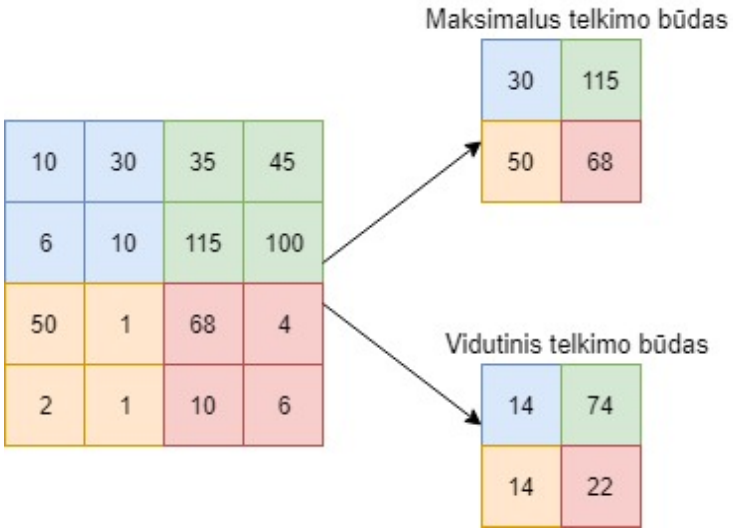
Konvoliucinis tinklas tai matematinė operacija, kuriai reikalingos dvi įvestys: vaizdo matrica ir filtras arba kernelis (angl. *kernel*) [25].

Kraštų aptikimo konvoliucinis skaičiavimas aprašytas skyreliuose 1.1, 1.2, kur kraštų aptikimui yra naudojami skirtingi tipo kerneliai.

1.7.2. Telkimo sluoksnis (angl. *Pooling layer*)

Panašiai kaip konvoliucinis sluoksnis, telkimo sluoksnis yra atsakingas už panašių ypatybių informacijos (angl. *Convolved Feature*) erdvinio dydžio mažinimą. Taip siekiama sumažinti skaičiavimus, reikalingus duomenims apdoroti. Be to, šio sluoksnio panaudojimas yra naudingas norint išgauti dominuojančias savybes, nekintančias sukimosi ar padėties prasme, taip išlaikant efektyvų modelio apmokymą.

Yra du telkimo tipai: maksimalus ir vidutinis. Naudojant maksimalų tipą, pateikiama didžiausia vertė iš atvaizdo dalies, kurią apima kernelis. Vidutinis tipas pateikia visų verčių vidurkį iš kernelio apimamos paveikslėlio dalies. Pavyzdys pateiktas 10 paveikslėlyje.



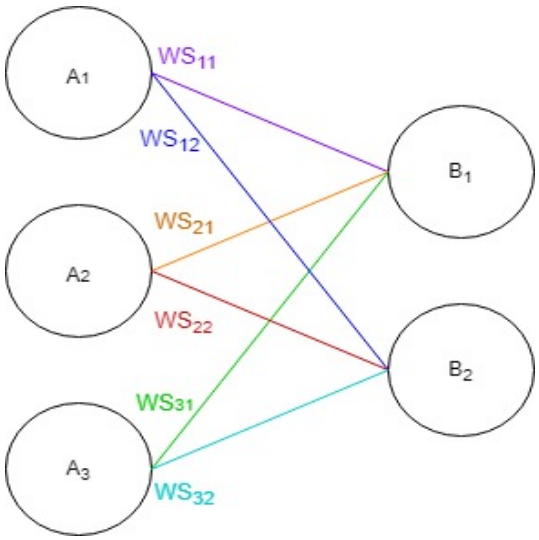
10 pav. Telkimo sluoksnio tipai.

Maksimalus telkimo tipas taip pat atlieka triukšmo slopinimą kartu su duomenų mažinimu. O vidutinis telkimo tipas tiesiog atlieka matmenų mažinimą kaip triukšmo slopinimo mechanizmas. Taigi galima teigti, kad šiuo atžvelgiu triukšmo ir duomenų mažinimo funkciją geriau atlieka maksimalus telkimo tipas.

[30]

1.7.3. Pilnai sujungtas sluoksnis (angl. *Fully-connected layer*)

Tai yra paprasčiausias sluoksnio tipas. Jis sujungia visas savo įvestis su visomis ankstesnio sluoksnio išvestimis.



11 pav. Visiškai sujungto sluoksnio su dviem neuronais pavyzdys.

Pav. 11 pavaizduotas visiškai sujungto sluoksnio su dviem neuronais pavyzdys, kur kiekviena įvestis pažymėta A_1, A_2 ir A_3, svoriai WS_{xx}, taigi išvestis B_1 ir B_2 galima suskaičiuoti su formule 1.12 [29]:

$$\begin{aligned} B_1 &= WS_{11} * A_1 + WS_{21} * A_2 + WS_{31} * A_3 \\ B_2 &= WS_{12} * A_1 + WS_{22} * A_2 + WS_{32} * A_3. \end{aligned} \tag{1.12}$$

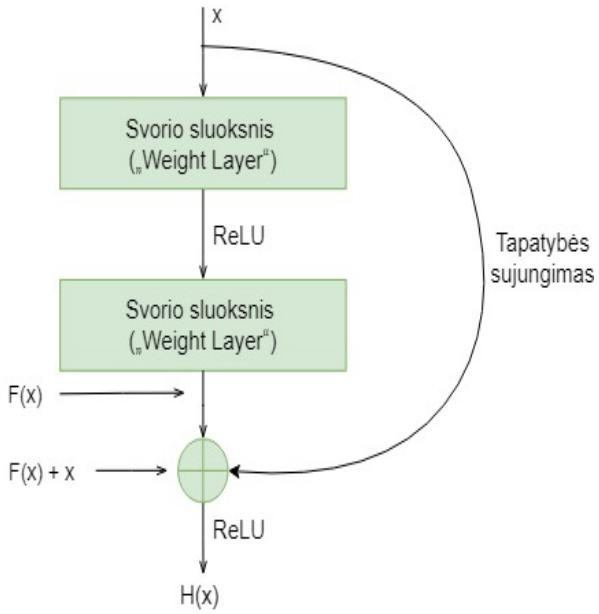
1.8. ResNet50 tinklas

Nuotraukos tiesinimo uždaviniui atlikti, buvo naudojamas RotNet modelis, kuris apmokymui naudoja ResNet50 modelį. ResNet50 modelis yra „ResNet“ modelio variantas, turintis 48 konvoliucinius sluoksnius ir po vieną maksimalų ir vidutinio telkimo sluoksnį [18].

„Deep Residual Network“ (ResNet) yra beveik panašus į tinklus, kuriuose vienas po kito eina konvoliuciniai, telkimo, suaktyvinimo ir visiškai sujungti sluoksniai. Vienintelė paprasto tinklo konstrukcija, paverčianti jį „išliekamuoju“ (angl. *residual*)) tinklu, yra tapatybės (angl. *identity*) sujungimas tarp sluoksnių. ResNet išliekamojo bloko pavyzdys parodytas pav. 12.

Išliekamoji funkcija (dar vadinama išliekamųjų suvedimas (angl. *residual mapping*)), pav. 12 žymima $F(x)$, yra skirtumas tarp nagrinėjamo išliekamojo bloko įvesties ir išvesties. Kitaip tariant, išliekamųjų suvedimas yra vertė, kuri bus pridėta prie įvesties, kad būtų galima apytiksliai įvertinti bloko galutinę funkciją (A_1, A_2, A_3, ... A_n). Taip pat galima teigti, jog išliekamųjų suvedimas yra klaidos, kurią galima pridėti prie įvesties, suma, kad būtų pasiektas galutinis tikslas, t. y. apytikslė galutinė funkcija. Išliekamųjų suvedimas veikia kaip tiltas tarp bloko įvesties ir išvesties. Aktyvavimo funkciją pav. 12 žymi $ReLU$, o blogo išvestis yra $H(x)$, kuri išsireiškiama formule 1.13, kur x yra bloko įvestis:

$$H(x) = F(x) + x \tag{1.13}$$



12 pav. ResNet išliekamasis blokas.

ResNet tinklo mokymo metu pagrindinis dėmesys skiriamas išliekamosios funkcijos, t.y. $F(x)$, išmokimui. Taigi, jei tinklas kažkaip išmoks skirtumą ($F(x)$) tarp įvesties ir išvesties, tada bendras tikslumus galimai padidės. Kitaip tariant, likutinė vertė turėtų būti išmokta taip, kad ji artėtų prie nulio, todėl išliekamųjų suvedimas taptų optimalus. Tokiu būdu visi tinklo sluoksniai visada sudarys optimalius ypatybių žemėlapius, t.y. geriausiu atveju ypatybių žemėlapį po konvoliucijos, sutelkimo ir aktyvavimo operacijų. Optimaliame ypatybių žemėlapyje yra visos svarbios savybės, kurios gali padėti klasifikuoti vaizdą [16].

Atgalinio skleidimo algoritmo, naudojamo apskaičiuojant kiekvieno svorio atnaujinimą per apmokymą ir paklaidos reikšmę, metu gradientai siunčiami atgal į vidinius sluoksnius ir atitinkamai atnaujinami svoriai. Gradientų nustatymo procesas ir jo siuntimas atgal į kitą vidinį sluoksnį tęsiamas tol, kol pasiekiamas įvesties sluoksnis. Gradientai tampa vis mažesni, kai procesas keliauja link tinklo pabaigos. Todėl pradinių sluoksnių svoriai arba labai lėtai atsinaujins, arba išliks tokie patys. Kitaip tariant, pradiniai tinklo sluoksniai neišmoks efektyviai. Vadinasi, gilaus tinklo mokymai nekonverguos, o tikslumas arba pradės blogėti, arba turės atitinkamą fiksuotą vertę. Nors nykstanti gradiento problema buvo sprendžiama naudojant normalizuotą svorių inicijavimą, gilesnio (angl. *deep*) tinklo tikslumas vis tiek nedidėjo.

ResNet tinkle yra du galimi keliai perduoti gradientų reikšmes atgal į įvesties sluoksnį. Diagramoje pav. 13, pirmasis kelias yra tapatybės sujungimo būdas, o atrasis - per išliekamųjų suvedimą. Matematiškai $F(x)$ galima išreikšti formule 1.14:

$$y = F(x, W_i) + x, \tag{1.14}$$

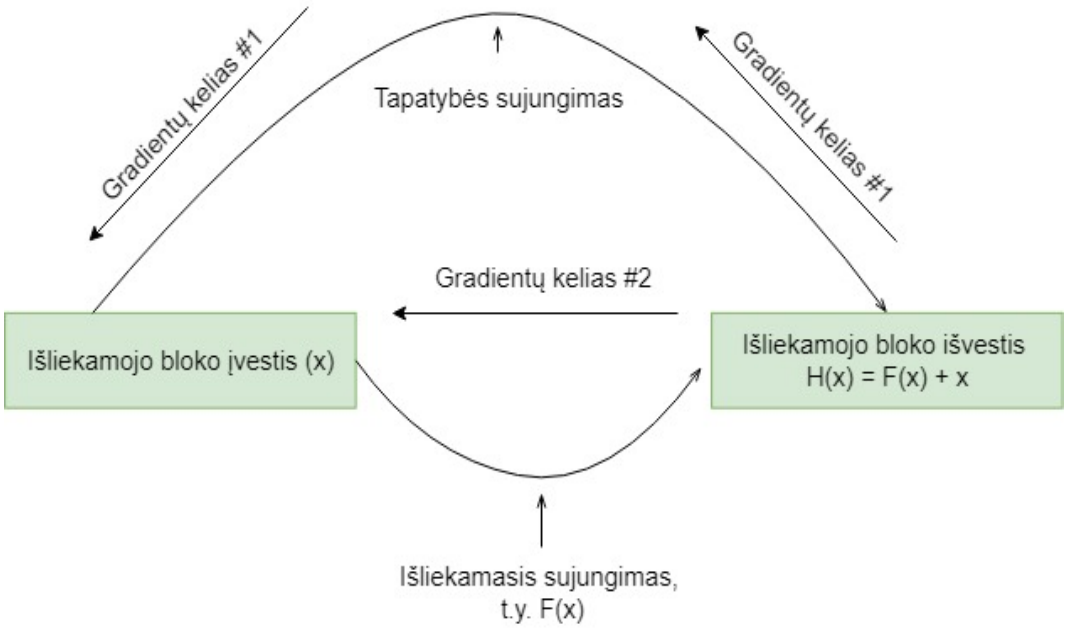
kur y yra funkcijos $F(x)$ išvestis, x - išliekamojo bloko įvestis, ir $F(x, W_i)$ yra išliekamasis blokas. Išliekamieji blokai sudaryti iš svorių sluoksnių, kurie žymimi W_i kur $1 \leqslant i \leqslant N$ - N žymi, kiek svorių sluoksnių yra išliekamajame bloke. $F(x, W_i)$, kai bloke yra 2 svorių sluoksniai, gali būti išreikštas formule 1.15:

$$F(x, W_i) = W_2\sigma(W_1x), \tag{1.15}$$

kur σ žymi aktyvavimo funkciją ReLU, po sudėjimo su tapatybės sujungimu (žr. diagramą 12), $H(x)$ išsireiškia formule 1.16:

$$H(x) = \sigma(y). \tag{1.16}$$

Kai suskaičiuoti gradientai perduodami iš antrojo kelio, yra susiduriama su dviem svorių sluoksniais - W_1 ir W_2 funkcijoje $F(x)$. Svorių reikšmės arba kerneliai yra atnaujinami svorių sluoksniuose W_1 ir W_2 ir apskaičiuojamos naujos gradientų vertės. Pradinių sluoksnių atveju naujai apskaičiuotos vertės arba sumažės, arba galiausiai išnyks. Taigi norint išsaugoti gradientų reikšmes nuo išnykimo, naudojamas spartusis ryšys (tapatybės sujungimas). Gradientai gali tiesiogiai pereiti per pirmąjį kelią, parodytą pav. 13. Pirmajame kelyje gradientai neturi susidurti su jokiu svorių sluoksniu, todėl apskaičiuotų gradientų vertė nepasikeis. Išliekamasis blokas bus iškart praleistas, gradientai gali pasiekti pradinius sluoksnius, kas padės jiems nustatyti teisingas svorių reikšmes.



13 pav. Gradientų perdavimo keliai ResNet išliekamajame bloke.

ResNet50 architektūrą sudaro 4 etapai. Tinklas gali imti įvesties vaizdą, kurio aukštis ir plotis yra padauginti iš 32 ir 3, kas atspindi spalvos kanalo plotį. Paaiškinimo sumetimais tarkime įvesties dydis yra $224 \times 224 \times 3$. Kiekviena „ResNet“ architektūra atlieka pradinę konvoliuciją ir maksimalų sujungimą naudodama atitinkamai 7×7 ir 3×3 dydžių kernelius. Vėliau prasideda 1 tinklo etapas ir jame yra 3 išliekamieji blokai, kuriuose yra po tris konvoliucinius sluoksnius. Kernelių dydis, naudojamas konvoliucijos operacijai atlikti visuose trijuose pirmos pakopos konvoliuciniuose sluoksniuose, yra atitinkamai 64, 64 ir 128. Tarp išliekamiejų blokų yra tapatybės sujungimas. Antro, trečio ir ketvirto etapo pirmoje pakopoje, konvoliucijos operacija išliekamajame bloke atliekama 2-jais žingsniais, taigi įvesties dydis bus sumažintas iki pusės pagal aukštį ir plotį, tačiau kanalo plotis bus padvigubintas. Kai pereinama iš vieno etapo į kitą, kanalo plotis padvigubėja, o įvesties dydis sumažėja iki pusės [28].

2. Kraštų ir linijų radimo algoritmų testavimo modelis

Šiame skyriuje pateikiamas testavimo modelio aprašas, skirtas rasti tinkamiausius kraštų ir linijų aptikimo algoritmus, programavimo indėlis ir pasiruošimas tyrimui atlikti naudojant tris kraštų aptikimo algoritmus: „Sobel“, „Scharr“ ir „Canny“ bei linijų radimo algoritmus: „Hough“ ir tikimybinę „Hough“ transformaciją. Indėlis į kodo rašymą pateiktas skyriuje 7.

2.1. Duomenys

Nuotraukos naudotos tyrimo metu yra iš asmeninio archyvo padarytos su fotoaparatu NIKON D5600. Paveiksliukai buvo sunumeruoti ir toliau tekste bus nurodomas jų numeris atitinkantis sąrašo 2.1 žemiau numerį, kuris aprašo paveiksliuko informaciją. Nuotraukų sąrašas:

1. Nuotrauka nr. 1: dydis: 773 KB, dimensija: 996 $\times$ 1375 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
2. Nuotrauka nr. 2: dydis: 896 KB, dimensija: 1000 $\times$ 1500 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
3. Nuotrauka nr. 3: dydis: 953 KB, dimensija: 1500 $\times$ 1000 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
4. Nuotrauka nr. 4: dydis: 1.34 MB, dimensija: 997 $\times$ 1500 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
5. Nuotrauka nr. 5: dydis: 1.37 MB, dimensija: 1000 $\times$ 1500 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
6. Nuotrauka nr. 6: dydis: 1.49 MB, dimensija: 1500 $\times$ 1000 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
7. Nuotrauka nr. 7: dydis: 674 KB, dimensija: 1500 $\times$ 1000 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.
8. Nuotrauka nr. 8: dydis: 1.09 MB, dimensija: 1500 $\times$ 1000 pikselių, horizontali / vertikali rezoliucija: 300 DPI, spalvos erdvė: sRGB.

2.2. Užtriukšminimas

Tyrimui buvo parinkti keletas skirtingų variantų triukšmui sugeneruoti: normalusis pasiskirstymas, druskos ir pipirų metodas (angl. *Salt and Pepper Noise*) [4]. Triukšmo tipų parinkimui šiam tyrimui įtakos turėjo perskaitytas straipsnis: Ruba Anas, Hadeel A. Elhadi ir Elmustafa Sayed Ali mini šiuos triukšmo algoritmus straipsnyje „Kraštų aptikimo algoritmų daroma įtaka medicinos nuotraukų apdorijimui“ (angl. *Impact of Edge Detection Algorithms in Medical Image Processing*) [1].

2.2.1. Normalusis skirstinys

Pridėtinio triukšmo pridėjimui naudojamas elementarus normalusis skirstinys (sutrumpinimas *NS*), sugeneruoti taškai, kurie pridėti prie originalios nuotraukos:
Pseodo kodas normalaus pasiskirstymo, kuomet paveiksliukas trijų RGB kanalų.

```
def noisy(image, mean, sigma):
    row,col,ch= image.shape
    gauss = np.random.normal(mean,sigma,(row,col,ch))
    gauss = gauss.reshape(row,col,ch)
    noisy = image + gauss
    return noisy
```

Funkcijai paduodama spalvota nuotrauka, grąžinama nuotrauka su pridėtiniu triukšmu. Tyrimui sugeneruotos dvi nuotraukos su pridėtiniu triukšmu: su intensyviu triukšmu, kai $\mu = 0$, o $\sigma = 60$, ir su nedideliu, tris kartus mažesniu triukšmu, kai $\mu = 0$, o $\sigma = 20$. Jeigu μ nelygu 0, pakeista nuotrauka per daug balta ir tai apsunkina kraštų radimo algoritmų testavimą, todėl μ abiem atvejais paliekamas 0. Gautas rezultatas po triukšmo uždėjimo parodytas paveikslėlyje 14, kur kairėje originalus paveiksliukas, paveiksliukas su intensyviu triukšmu, ir tas pats paveiksliukas su tris kartus mažesne σ reikšme.

14 pav. Triukšmo uždėjimo gautas rezultatas: normaliojo skirstinio metodas. Nuotrauka nr. 1.

2.2.2. Druskos ir pipirų metodas - pirmas sprendimas

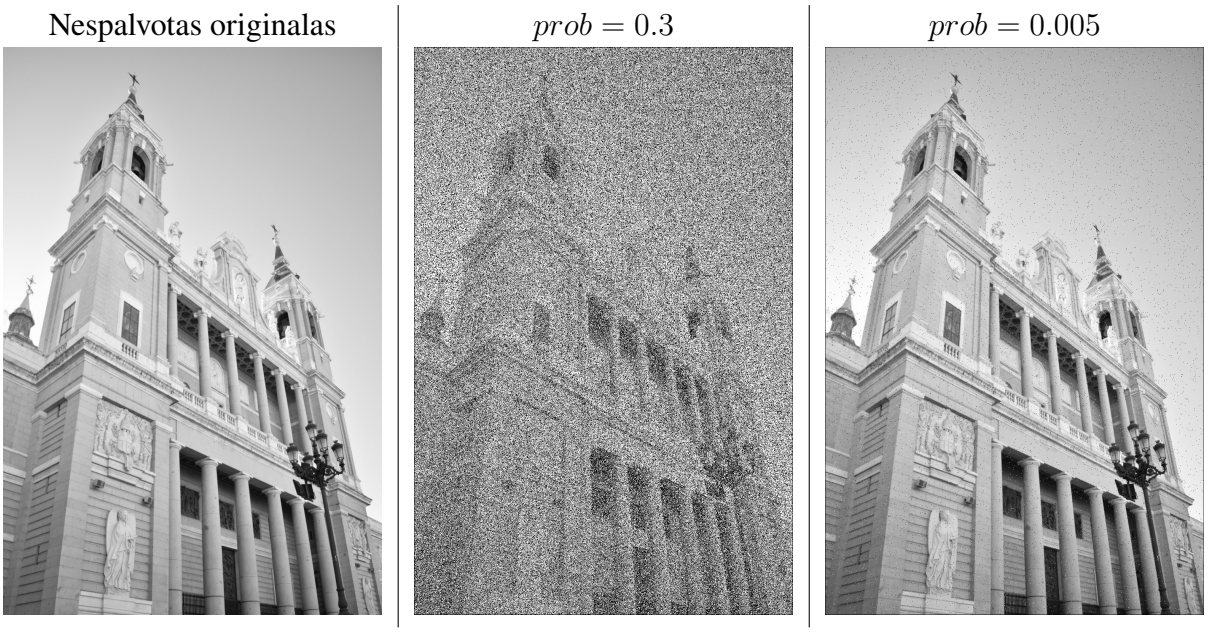
Druskos ir pipirų (sutrumpinimas *DIP*) metodo metu, nuotraukos pikseliai, kurie nepatenka tarp ribos žymų, pakeičiami juodais arba baltais (nustatoma 0 arba 255 reikšmė). Pikselio pakeitimo funkcija $f(i, j)$ apibrėžta formulėje 2.1, kur AR - apatinė riba, VR - viršutinė riba, $I(i, j)$ -

originalaus paveiksliuko pikselis.

$$f(i,j) = \begin{cases} 0, & \text{jeigu } I(i,j) < \text{AR} \\ 255, & \text{jeigu } I(i,j) > \text{VR} \\ I(i,j), & \text{kitu atveju} \end{cases} \tag{2.1}$$

Pirmas sprendimas, kuomet funkcijai generuojančiai šio tipo triukšmą, paduodama tikimybės reikšmė $prob$, parenkamas atsitiktinis skaičius, ir jeigu jis mažesnis už $prob$ reikšmę, tuomet pikselio reišmė prilyginima 0, jeigu didesnis už $1 - prob$, tuomet pikselis tampa baltu (255 reikšmė), kitu atveju priskiriamas nespalvotos nuotraukos pikselis [4]. Gautas rezultatas, kai $prob = 0.3$ ir $prob = 0.005$ pavaizduotas paveikslėlyje 15. Funkcija priima nespalvotą nuotrauką ir grąžina nuotrauką su pridėtu triukšmu, pseodo kodas pateiktas žemiau:
Druskos ir pipirų metodo pseodo kodas - pirmas sprendimas, kuomet paveiksliukas nespalvotas.

```
def saltAndPepper_noiseFirst(grayImage, prob):
    noisy = np.zeros(grayImage.shape,np.uint8)
    threshold = 1 - prob
    h, w = grayImage.shape
    for i in range(1, h - 1):
        for j in range(1, w - 1):
            random = rn.random()
            if random < prob:
                noisy[i][j] = 0
            elif random > threshold:
                noisy[i][j] = 255
            else:
                noisy[i][j] = grayImage[i][j]
    return noisy
```



15 pav. Triukšmo uždėjimo gautas rezultatas: druskos ir pipirų metodas, pirmas sprendimas. Nuotrauka nr. 1.

2.2.3. Druskos ir pipirų metodas - antras sprendimas

Antras sprendimas, kuomet funkcijai generuojančiai šio tipo triukšmą, paduodama tikimybės reikšmė σ ir ribos sugeneruotos remiantis ta pačia logika, kaip ir ribos „Canny“ algoritmui aprašytame skyrelyje 1.4, ribų funkcijos išreikštos formulėje 1.10. Jeigu originalios nuotraukos pikeselis mažesnis už $lower$ ribos reikšmę, tuomet pikselio reišmė prilyginima 0, jeigu didesnis už $upper$ ribos reikšmę, tuomet pikselis tampa baltu (255 reikšmė). Dėl šios logikos, kuo **mažesnė σ reikšmė, tuo daugiau pikselių bus pakeista (užtriukšminta)**. Gautas rezultatas, kai $sigma = 0.005$ ir $\sigma = 0.33$ pavaizduotas pav. 16. Funkcija priima nespalvotą nuotrauką ir grąžina nuotrauką su pridėtu triukšmu, pseodo kodas pateiktas žemiau:
Druskos ir pipirų metodo pseodo kodas - antras sprendimas, kuomet paveiksliukas nespalvotas.

```
def saltAndPepper_noiseSecond(grayImage, sigma):
    noisy = np.zeros(grayImage.shape,np.uint8)
    v = np.median(image)
    lower = int(max(0, (1.0 - sigma) * v))
    upper = int(min(255, (1.0 + sigma) * v))
    h, w = grayImage.shape
    for i in range(1, h - 1):
        for j in range(1, w - 1):
            if grayImage[i][j] < lower:
                noisy[i][j] = 0
            elif grayImage[i][j] > upper:
                noisy[i][j] = 255
            else:
                noisy[i][j] = grayImage[i][j]
    return noisy
```

16 pav. Triukšmo uždėjimo gautas rezultatas: druskos ir pipirų metodas, antras sprendimas. Nuotrauka nr. 1.

2.3. Kraštų aptikimo algoritmai

Visi trys tiriami algoritmai geriausiai randa kraštus, kuomet nuotrauka yra nespalvota, verta paminėti, kad tokiu būdu yra sumažinamas ir triukšmo lygis nuotraukoje - nespalvota nuotrauka turi žymiai mažiau triukšmo negu spalvota, nes algoritmas skaičiuoja tik vieno kanalo (angl. *channel*) lygmenyje, ir pikselių vertės nuo 0 iki 255, taigi pirmas veiksmas bus paversti nuotrauką į nespalvotą [31]. Nespalvotą nuotrauką versime su „OpenCV“ funkcijos pagalba: *cv2.cvtColor(imageOriginal, cv2.COLOR_BGR2GRAY)* [8].

Kadangi testavimui naudosime „Canny“ algoritmą iš „OpenCV“ bibliotekos, tyrimui atlikti buvo parašytas algoritmų „Sobel“ ir „Scharr“ kodas. Kadangi operatoriai panašūs, parašyta funkcija priima du parametrus: nespalvotą nuotrauką ir algoritmo tipą - „Sobel“ arba „Scharr“, kitu atveju funkcija išmes klaidos pranešimą. Funkcija grąžina tris gradientų rinkinius: horizontalius, vertikalius ir susumuotus (visus) kraštus.

Tyrimui naudotas „Canny“ algoritmas iš „OpenCV“ bibliotekos, patobulintas prieš tai parinktus tinkamus parametrus, plačiau apie „Canny“ parametrų parinkimą poskyryje 1.4. Funkcija priima nespalvotą nuotrauką ir grąžina rastų kraštų rinkinį.

2.4. Linijų aptikimo algoritmai

Linijų radimui tyrime naudoti du algoritmai: „Hough“ transformacijos ir tikimybinės „Hough“ transformacijos algoritmai iš „OpenCV“ bibliotekos. Tyrimui atlikti buvo implementuotos dvi funkcijos priklausomai nuo „Hough“ transformacijos (tikimybinė ar ne) tipo, funkcijų esmė rasti optimaliausius „Hough“ algoritmų parametrus su kuriais būta rasta linijų originalioje ar labai užtriukšmintoje nuotraukoje ir uždėti linijas ant originalaus paveiksliuko.

Pradinės „Hough“ transformaciją naudojančios funkcijos parametras yra tik vienas - riba, kuri nustatyta 250, o tikimybinę „Hough“ transformaciją naudojančios funkcijos pradiniai parametrai: riba lygi 100, *„minLineLength“* lygu 100, *„maxLineGap“* lygu 10 remiantis „OpenCV“ rekomendacija [10]. Jeigu su pagal numatymą nustatytomis pradinėmis parametrų reikšmėmis linijų nerasta, parametrų reikšmės švelninamos 10%. Pagal vienetą buvo pasirinkta nešvelninti, nes skaičiavimai užtruktų, o darbui atlikti užtenka rasti keletą linijų, be to, rezultatai tokie patys kaip mažinant per vieną vienetą, pavyzdys lentelėse 10 ir 11.

Funkcijos metu uždedama pirmos 20 ilgiausios rastos linijos (stipriausios linijos). Funkcijos priima tokius parametrus: originalią nuotrauką ant kurios bus uždedamos linijos, gradientų (kraštų) rinkinį. Funkcijos grąžina originalią nuotrauką su uždėtomis linijomis (linijų spalva - raudona, plotis 4 pikseliai). Šios funkcijos buvo pavadintos *linedPictureHoughLinesBestParametersChange10Perc* ir *linedPictureHoughLinesPBestParametersChange10Perc*.

Po šių funkcijų pritaikymų pastebėta, jog po pirmo testo, „Sobel“ ir „Scharr“ algoritmai atkrenta tolimesnei analizei , randama daug įstrižų bei vertikalių linijų po „Canny“ algoritmo pritaikymo ir DIP sugeneruoja triukšmą dėl kurio randama nuotraukos kraštai. Pavyzdys tokių linijų atvaizduotas po šios funkcijos pav. 17.

DIP pirmas spr. po antro testo | DIP pirmas spr. po trečio testo

17 pav. Įstrižų/vertikalių ir nuotraukos kraštus atitinkančių linijų pavyzdys po antro ir trečio testo. Druskos ir pipirų metodas, pirmas sprendimas, „Hough“ transformacija. Nuotrauka nr. 1.

Dėl to tolimesniems testams (visiems išskyrus pirmą), buvo naudotos patobulintos funkcijos *linedPictureHoughLinesBestParametersChange* ir *linedPictureHoughLinesPBestParametersChange*. Patobulinta „Hough“ transformacijos linijų uždėjimo dalis - tikrinama, ar gauto kampo θ reikšmė patenka tarp 0 ir 45 laipsnių. Tas pats nebuvo padaryta tikimybinės „Hough“ transformacijos funkcijoje, nes ši funkcija negrąžina kampo reikšmės.

Abiejų linijų radimo ir uždėjimo funkcijų metu, buvo realizuotas linijų atmetimas, kurios arčiau nei 5 pikseliai prie nuotraukos krašto (uždedamos linijos dydis 4 pikseliai, todėl buvo nustatyta atmesti kas arčiau 5 pikselių) - toks sprendimas buvo priimtas pamačius, jog po „Canny“ operatoriaus praleidimo su nuotrauka, kuriai uždėtas DIP tipo triukšmas, gauta linijos einančios aplink nuotraukos kraštus (rėmą).

Pradiniai parametrai buvo nustatyti tokie:

- „Hough“ transformacijoje riba 250, kritinė reikšmė 100 (pagal „OpenCV“ dokumentaciją 100 riba taikoma tikimybinei „Hough“ transformacijoje (švelni reikšmė) [10].
- Tik. „Hough“ transformacijoje riba ir minimalus linijos ilgis 250, kritinės reikšmės 100, maksimalus tarpas - 40 (padidinta 4 kartus negu rekomenduojamas, todėl kad tyrime analizuojamų paveiksliukų dimensija yra apie 3-4 kartus didesnis negu „OpenCV“ pavyzdžiuose), kritinė reikšmė lygi 190 (kadangi ribą ir minimalios linijos ilgį daugiausia mažinsim per 150 vienetų).

Abiejų algoritmų metu tiriama pirmos 40 grąžinamų linijų. Funkcijos grąžina „Linijų nerado!“ paveiksliuką, jeigu:

1. Pasiektos visos kritinės parametrų reikšmės ir dėl to linijų nerado visai.
2. Jeigu pirmos 40 linijų neatitiko kampo kriterijaus (tarp 0 ir 45 laipsnių) po „Hough“ transformacijos.
3. Jeigu pirmos 40 linijų yra arčiau nei 5 pikseliai nuo nuotraukos krašto.

Pavyzdys, kas gauta po patobulinto algoritmo pavaizduotas pav. 18, kur skirtingai, nei pavyzdyje 17, galima pamatyti, jog po antro testo neliko vertikalių linijų, o po trečio testo linijos nepraėjo kriterijų, todėl gautas „Linijų nerado!“ paveiksliukas.

DIP pirmas spr. po antro testo

DIP pirmas spr. po trečio testo

Linijų nerado!

18 pav. Rezultatai antro ir trečio testo. Druskos ir pipirų metodas, pirmas sprendimas, patobulinta „Hough“ transformacijos ir linijų uždėjimo funkcija. Nuotrauka nr. 1.

2.5. Testavimo planas

Testavimo metu yra naudojamos tik spalvotos nuotraukos paverčiamos į nespalvotas. Testavimo planą sudaro:

1. Pirmo testo metu originali nespalvota nuotrauka praleidžiama pro visus tris darbe analizuojamus kraštų aptikimo algoritmus. Po šio testo, randamos linijos su paprasta ir tikimybine „Hough“ transformacija.

2. Antro testo metu originali nespalvota nuotrauka, kuriai pridėta mažesnis kiekis triukšmo, praleidžiama pro visus tris kraštų aptikimo algoritmus. Po šio testo, randamos linijos.

3. Trečio testo metu originali nespalvota nuotrauka, kuriai pridėtas intensyvus kiekis triukšmo, praleidžiama pro visus tris kraštų aptikimo algoritmus. Po šio testo, randamos linijos.

4. Ketvirto testo metu originali nespalvota nuotrauka, kuriai pridėtas intensyvus kiekis triukšmo, yra nutriukšminama naudojant Gauso filtro funkciją, aprašytą poskyryje 1.3.1, su didžiausią suliejimą padariusiais parametrais, kuomet kernelio dydis 7, o $\sigma = 60$, pavyzdys suliejimo pav. 3. Po nutriukšminimo, gauta susiliejusi nuotrauka praleidžiama pro visus tris kraštų aptikimo algoritmus. Po šio testo, randamos linijos su paprasta ir tikimybine „Hough“ transformacija.

3. Kraštų aptikimo ir linijų radimo tyrimas

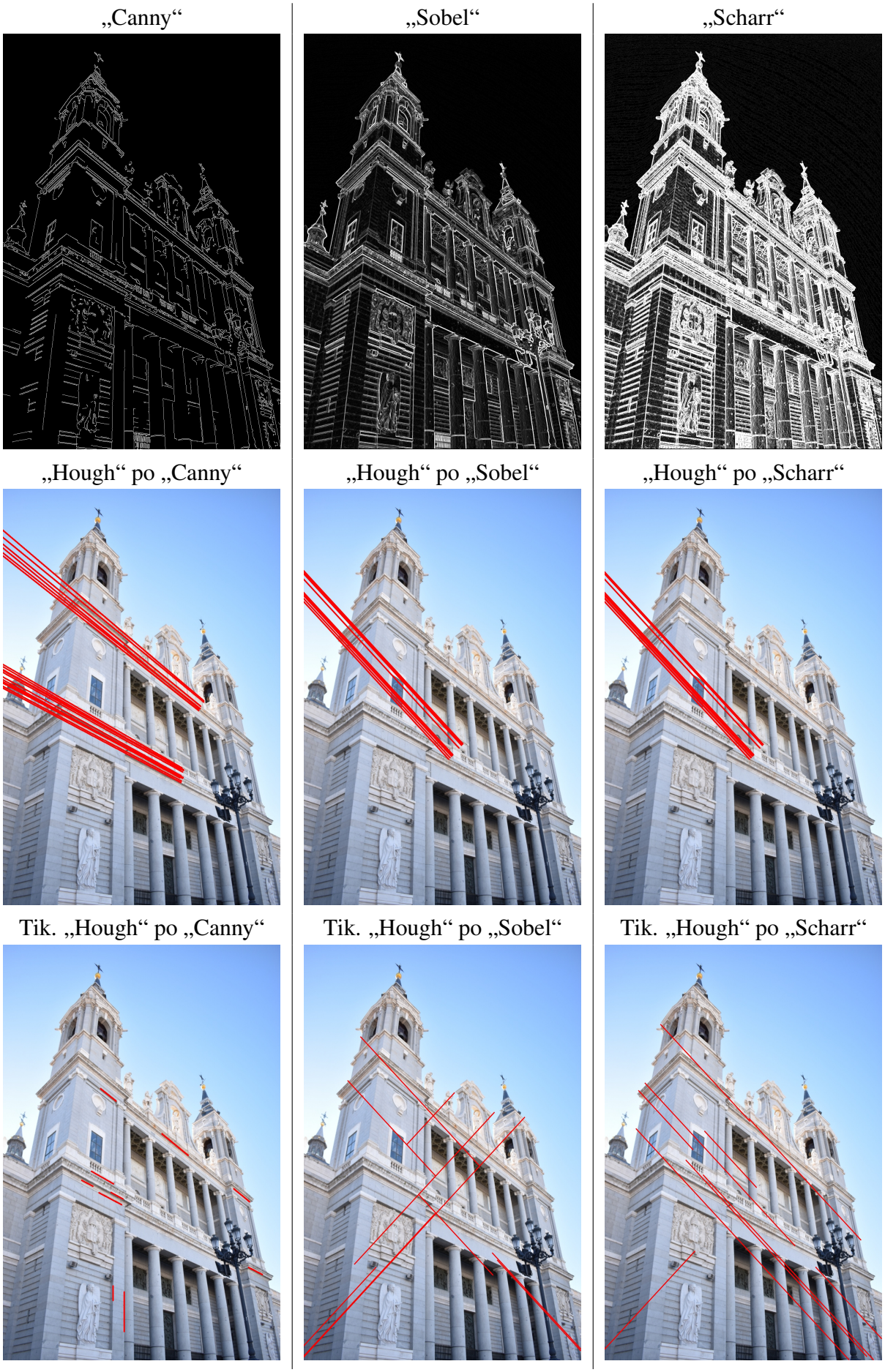
Šiame skyriuje aprašytas tyrimas, kurio planas nurodytas poskyryje 2.5. Žemiau pateikti paveiksliukai su tyrimui paruošta nuotrauka nr. 1.

3.1. Pirmas testas - su originaliomis nuotraukomis

Pirmo testo rezultatai gauti praleidus originalų paveiksliuką per tris analizuojamus kraštų aptikimo algoritmus ir gautus kraštus pateikiant „Hough“ transformacijos ir tikimybinės „Hough“ transformacijos funkcijoms. Iš gauto rezultato, kuris atvaizduotas pav. 19, galima daryti išvadą, kad „Sobel“ ir „Scharr“ algoritmai netinkami linijų paieškai, naudojant „Hough“ transformaciją. Gautos linijos nesutampa su vizualiai matomomis pastato kontūro linijomis analizuojamoje nuotraukoje. Netgi pritaikius Gauso filtrą originaliai nuotraukai, rezultatai tokie patys, pvz.: pav. 31.

Gauti linijų kampai yra apie 45 laipsnius ir daugiau pasvirę nuo horizonto (įstrižinės linijos), taigi praleidus pro patobulintas funkciją *linedPictureHoughLinesBestParametersChange*, gautas rezultatas yra „Linijų nerado!“ paveiksliukas, nes 45 ir daugiau laipsnių linijos yra atmetamos. Taigi, po pirmo testo, „Sobel“ ir „Scharr“ algoritmai atkrenta tolimesnei analizei.

Po pirmo testo, naudojant „Canny“ algoritmo rastus originalios nuotraukos kraštus, paprasta ir tikimybinė „Hough“ transformacijos rado linijų, iš gauto rezultato galima matyti, kad linijos atitinka pastato horizontalius kontūrus. „Canny“ algoritmo stiprioji pusė, jog radus kraštus jis atlieka papildomus glotninimo ir nereikšmingų taškų atmetimo žingsnius, tokius kaip ne-maksimumo slopinimas (žr. skyrelį 1.3.3), dvigubos ribos (žr. skyrelį 1.3.4) ir histerezės panaudojimo (žr. skyrelį 1.3.5).



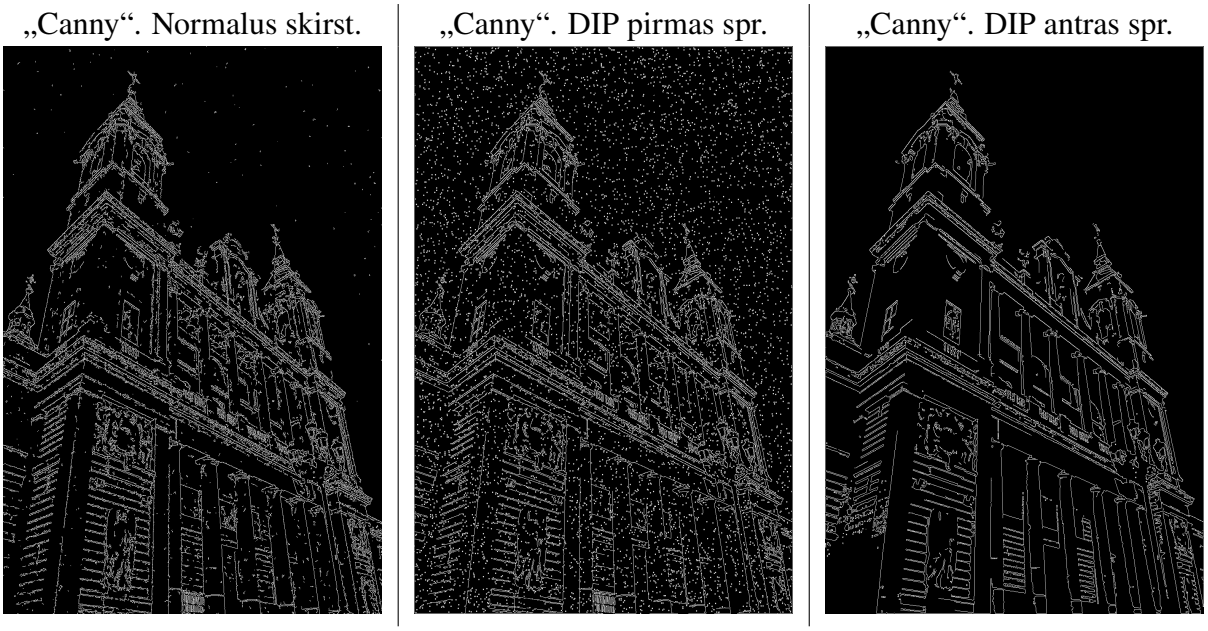
19 pav. Pirmo testo rezultatai. Nuotrauka Nr. 1.

3.2. Antras testas - su pridėtiniu triukšmu

Šiame skyrelyje pateikiami gauti rezultatai po antro testo, naudojant tik „Canny“ kraštų aptikimo algoritmą, kai triukšmui sugeneruoti naudoti metodai:

1. Normalusis skirstinys (NS), kai $\mu = 0$, o $\sigma = 20$.
2. Druskos ir pipirų metodas (DIP) - pirmas sprendimas, kai $prob = 0.005$.
3. Druskos ir pipirų metodas (DIP) - antras sprendimas, kai $\sigma = 0.33$.

Žemiau pav. 20 pateikiamas palyginimas, ką grąžino „Canny“ kraštų aptikimo algoritmas po pritaikytų skirtingų triukšmo metodų.

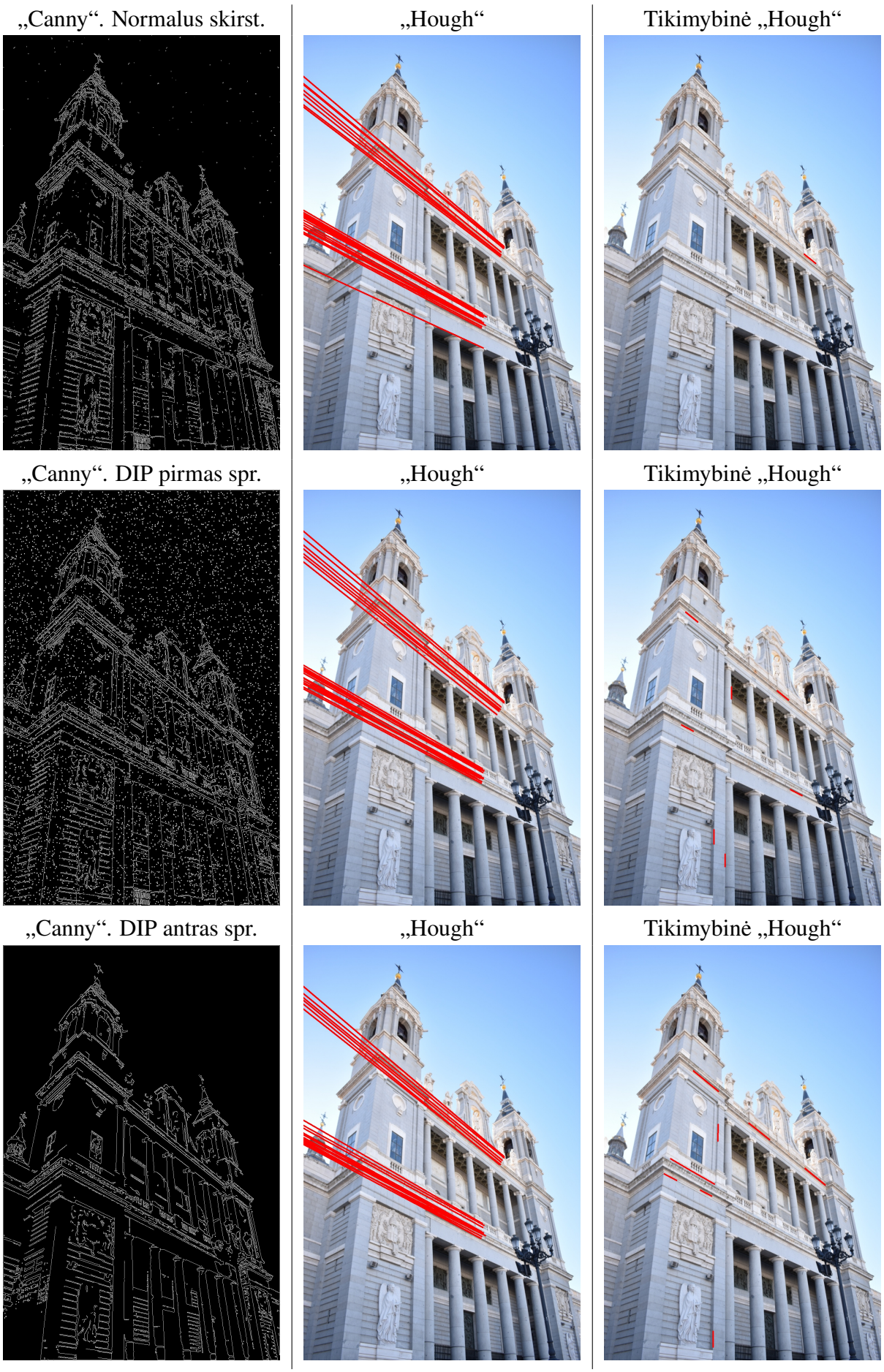


20 pav. „Canny“ kraštų aptikimo algoritmo rezultatas po pritaikytų skirtingų triukšmo metodų. Nuotrauka Nr. 1.

Antro testo metu, naudojant „Canny“ algoritmo rastus užtriukšmintos nuotraukos kraštus, „Hough“ transformacijos funkcija rado linijų, iš gauto rezultato pav. 21 galima matyti, kad horizontalios linijos atitinka pastato horizontalius kontūrus. Iš gautų rezultatų 1 lentelės matoma, kad normalio skirstinio triukšmo metu „Hough“ transformacija rado 5 linijom daugiau, o tikimybinė - 8 linijom mažiau nei originalioje linijoje. Rastų linijų kampų reikšmės apytiksliai tokios pačios. Galima teigti, kad triukšmas nebuvo didelis ir didelės įtakos linijų paieškai nepadarė.

1 lentelė. Antro testo parametrų rezultatai su nuotrauka Nr. 1., remiantis linijų radimo parametrų rezultatais po pirmo testo. Kraštų aptikimo algoritmas - „Canny“.

	„Hough“			Tik. „Hough“			
Testo nr. ir triukšmo tipas	Linijų sk.	Riba	Kampas (radianais)	Linijų sk.	Riba	Min. linijos ilgis	Maks. tarpas
1.	17	250	2.024582 - 2.2340214	10	250	250	40
2. NS	25	250	2.0071287 - 2.2863812	2	250	250	40
2. DIP 1 spr.	18	250	2.024582 - 2.2863812	10	250	250	40
2. DIP 2 spr.	17	250	2.0071287 - 2.2340214	9	250	250	40



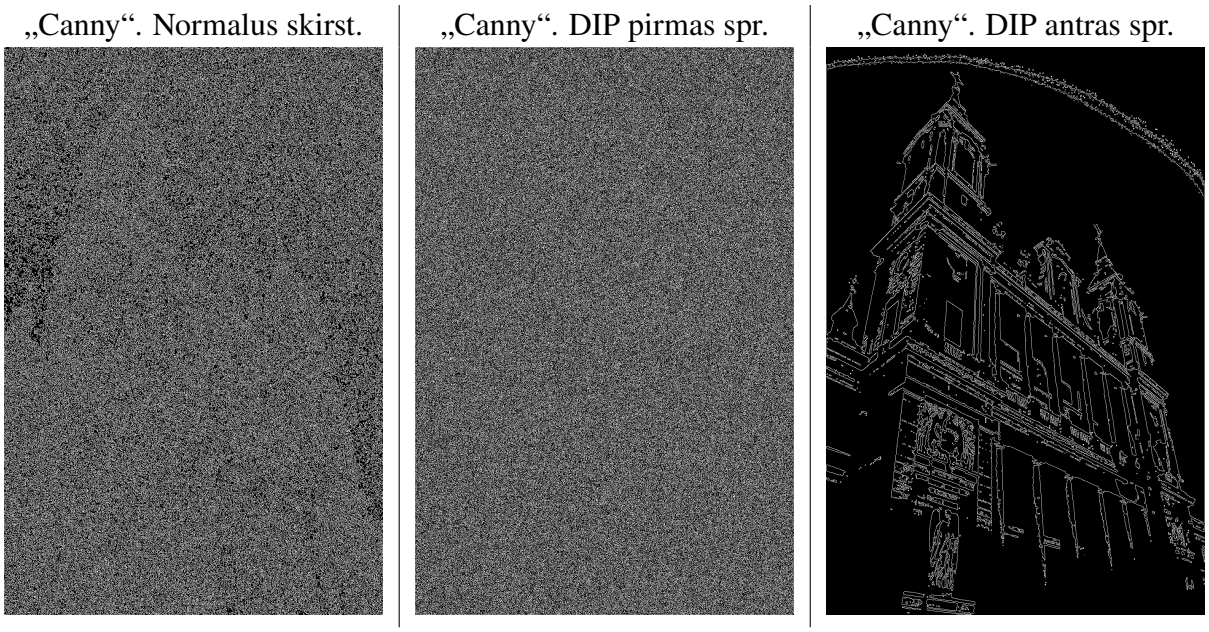
21 pav. Antro testo rezultatai - linijų radimas naudojant „Canny“, triukšmo tipas: normalus skirstinys, DIP pirmas ir antras sprendimai. Nuotrauka Nr. 1.

3.3. Trečias testas - su didesniu kiekiu pridėtinio triukšmo

Šiame skyrelyje pateikiami gauti rezultatai po trečio testo, naudojant tik „Canny“ kraštų aptikimo, kai triukšmui sugeneruoti naudoti metodai:

1. Normalusis skirstinys, kai $\mu = 0$, o $\sigma = 60$.
2. Druskos ir pipirų metodas - pirmas sprendimas, kai $prob = 0.3$.
3. Druskos ir pipirų metodas - antras sprendimas, kai $\sigma = 0.005$.

Žemiau pav. 22 pateikiamas palyginimas, ką grąžino „Canny“ kraštų aptikimo algoritmas po pritaikytų skirtingų triukšmo metodų. Galima pamatyti, jog normaliojo skirstinio ir DIP pirmo sprendimo algoritmai visiškai sugadino nuotrauką, nėra įmatomi jokie kraštai, tuo tarpu po DIP antras sprendimo algoritmo pritaikymo gauti kraštai labai smulkmeniški, atsirado dangaus skliauto linija, tačiau kraštai įželgiami, vizualiai sprendžiant, drąstiškos įtakos kraštų radimui šis triukšmo generavimo sprendimas nepadarė.



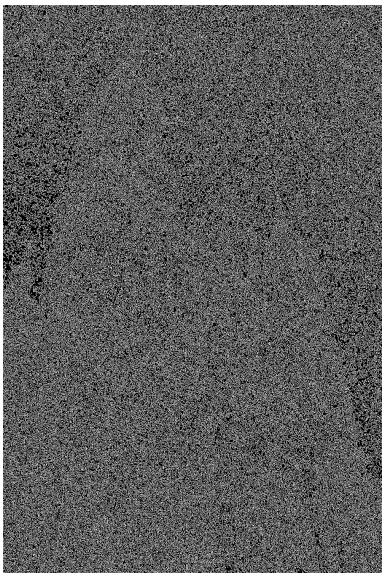
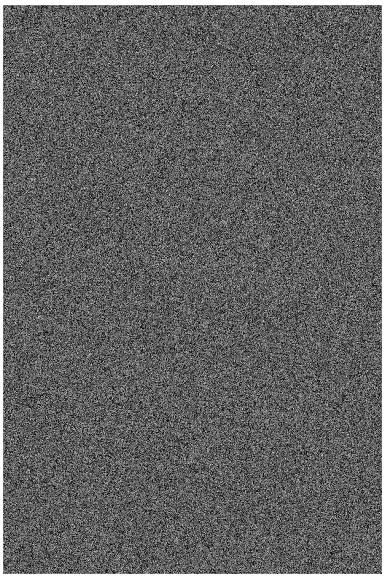
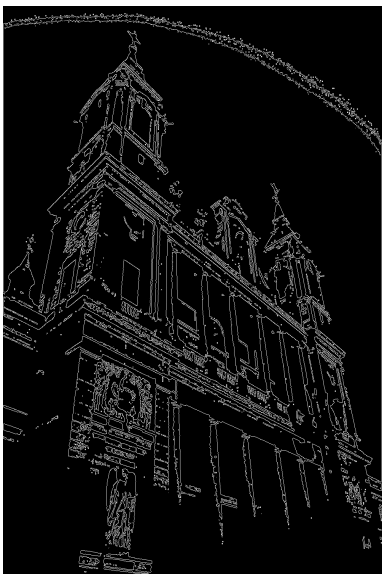
22 pav. „Canny“ kraštų aptikimo algoritmo rezultatas po pritaikytų skirtingų triukšmo metodų. Nuotrauka Nr. 1.

Trečio testo metu, naudojant „Canny“ algoritmo rastus užtriukšmintos nuotraukos kraštus, paprastos bei tikimybinės „Hough“ transformacijos funkcijos nerado linijų su normaliojo skirstinio ir DIP pirmo sprendimo būdu užtriukšmintomis nuotraukomis. Rezultatai matomi pav. 23.

Linijos buvo rastos su abiems „Hough“ transformacijos algoritmais tik tuo atveju, kai buvo užtriukšminta su DIP antru sprendimo būdu ir dauguma rastų linijų atitinka vietas, kuriose rado linijų „Hough“ algoritmai pirmo testo metu, be to rastos kampų reikšmės yra tarp 2.024582 - 2.2340214, kaip ir pirmo testo metu. Linijų rasta mažiau. Galima teigti, kad tik DIP antro sprendimo atveju sugadintoje nuotraukoje pavyko rasti linijų.

2 lentelė. Trečio testo parametrų rezultatai su nuotrauka Nr. 1., remiantis linijų radimo parametrų rezultatais po pirmo testo. Kraštų aptikimo algoritmas - „Canny“.

	„Hough“			Tik. „Hough“			
Testo nr. ir triukšmo tipas	Linijų sk.	Riba	Kampas (radianais)	Linijų sk.	Riba	Min. linijos ilgis	Maks. tarpas
1.	17	250	2.024582 - 2.2340214	10	250	250	40
3. NS	0	250	✗	0	✗	✗	✗
3. DIP 1 spr.	0	250	✗	0	✗	✗	✗
3. DIP 2 spr.	11	250	2.024582 - 2.2340214	3	250	250	40

„Canny“. Normalus skirst.	„Hough“	Tikimybinė „Hough“
	Linijų nerado!	Linijų nerado!
„Canny“. DIP pirmas spr.	„Hough“	Tikimybinė „Hough“
	Linijų nerado!	Linijų nerado!
„Canny“. DIP antras spr.	„Hough“	Tikimybinė „Hough“
		

23 pav. Trečio testo rezultatai - linijų radimas naudojant „Canny“, triukšmo tipas: normalus skirstinys, DIP pirmas ir antras sprendimai. Nuotrauka Nr. 1.

3.4. Ketvirtas testas - Gauso Filtro panaudojimas

Ketvirto testo metu originali nespalvota nuotrauka, kuriai pridėtas intensyvus kiekis triukšmo su triukšmo pridėjimo metodais aprašytais trečio testo apžvalgos skyrelyje 3.3, yra nutriukšminama naudojant Gauso filtro funkciją, aprašytą poskyryje 1.3.1, su didžiausią suliejimą padariusiais parametrais, kuomet kernelio dydis 7, o $\sigma = 60$, ir tik tada nuotraukai pritaikomas „Canny“ kraštų aptikimo algoritmas ir ieškoma linijų.

Gauti testo rezultatai pavaizduoti pav. 24. Iš šio pav. ir 3 lentelės galima teigti, kad Gauso filtras gerokai sumažino linijų skaičių ir yra per daug agresyvus, todėl paprastos bei tikimybinės „Hough“ transformacijos funkcijos neuždėjo linijų ant nuotraukų, kurios užtriukšmintos normaliojo skirstinio (šiuo atveju buvo pasiekta kritinė riba ieškant linijų!) ir DIP pirmo sprendimo būdu. Linijos rastos, kai buvo užtriukšminta su DIP antru sprendimo būdu, rezultatai beveik tokie patys kaip ir prieš tai buvusių testų.

3 lentelė. Ketvirto testo parametrų rezultatai su nuotrauka Nr. 1., remiantis linijų radimo parametrų rezultatais po pirmo testo. Kraštų aptikimo algoritmas - „Canny“.

	„Hough“			Tik. „Hough“			
Testo nr. ir triukšmo tipas	Linijų sk.	Riba	Kampas (radianais)	Linijų sk.	Riba	Min. linijos ilgis	Maks. tarpas
1.	17	250	2.024582 - 2.2340214	10	250	250	40
4. NS	0	✗	✗	0	✗	✗	✗
4. DIP 1 spr.	0	250	✗	0	✗	✗	✗
4. DIP 2 spr.	10	250	2.0071287 - 2.2165682	4	250	250	40

Po ketvirto testo, galima teigti, kad reikia penkto testo! Penkto neoficialaus testo metu bus sumažinti Gauso Filtro parametrai, $K = 5$, o $\sigma = 0$ (mažiau suliejama nuotrauka). Sumažinus Gauso filtro parametrus, iš pav. 25 ir 4 lentelės pateiktų rezultatų, galima pamatyti, jog „Hough“ transformacija rado linijų su ribos reikšme lygia 250, kai triukšmo tipas normalusis skirstinys, tik dvi linijos praėjo kriterijus (dėl šitos priežasties nerasta linijų su tikimybine „Hough“ transformacija, jos metu dauguma taškų buvo pašalinta ir su minimaliom reikšmėmis nerado linijų). Rastų linijų kampas patenka į kampų reikšmių sąrašą iš pirmo testo, taigi šio testo metu sėkmingai rasta linijų, kai triukšmo tipas normalusis skirtinys ir DIP antras sprendimas. Tuo tarpu, DIP pirmo sprendimo metu sugeneruotas triukšmas visiškai sugadino nuotrauką, ir penkto testo metu linijų uždėta nebuvo.

4 lentelė. Penkto testo parametrų rezultatai su nuotrauka Nr. 1., remiantis linijų radimo parametrų rezultatais po pirmo testo. Kraštų aptikimo algoritmas - „Canny“.

	„Hough“			Tik. „Hough“			
Testo nr. ir triukšmo tipas	Linijų sk.	Riba	Kampas (radianais)	Linijų sk.	Riba	Min. linijos ilgis	Maks. tarpas
1.	17	250	2.024582 - 2.2340214	10	250	250	40
5. NS	2	250	2.024582 - 2.042035	0	✗	✗	✗
5. DIP 1 spr.	0	250	✗	0	✗	✗	✗
5. DIP 2 spr.	25	250	2.0071287 - 2.2863812	9	250	250	40

Po šių testų galima sudaryti rezultatų lentelę 5, kuomet buvo uždėtos linijos sprendžiant iš prieš tai gautų parametrų rezultatų lentelių. Iš šios lentelėje pavaizduotų rezultatų galima daryti išvadą, kad DIP antro sprendimo metu sugeneruotas triukšmas linijų radimui įtakos nepadarė, tuo tarpu DIP pirmo sprendimo metu sugeneruotas intensyvus triukšmas visiškai sugadino nuotrauką. Po normaliojo pasiskirstymo metu sugeneruoto triukšmo, linijas pavyko rasti tik sušvelninus Gauso filtro parametrus, kitu atveju buvo pašalinta per daug kraštų - nuotrauka savaime nebuvo labai „triukšminga“, t.y. turinti daug smulkių objektų.

5 lentelė. Testų rezultatai su nuotrauka Nr. 1., remiantis, ar linijos rastos iš lentelių 1, 2, 3 ir 4 duomenų. Kraštų aptikimo algoritmas - „Canny“.

Testo numeris	„Hough“			Tik. „Hough“		
1	✓			✓		
	Norm. skir.	DIP 1 spr.	DIP 2 spr.	Norm. skir.	DIP 1 spr.	DIP 2 spr.
2	✓	✓	✓	✓	✓	✓
3	✗	✗	✓	✗	✗	✓
4	✗	✗	✓	✗	✗	✓
5 (neoficialus)	✓	✗	✓	✗	✗	✓

Tuo tarpu paėmus labiau savaime „triukšmingą“ nuotrauką, pvz. nuotrauka nr. 5 turi labai daug smulkių objektų, „Hough“ transformacijos atveju gauta daug įstrižų linijų, tikimybinės atveju gauta smulkių linijų. Praleidus per patobulintus linijų uždėjimo algoritmus gautas rezultatas pavaizduotas pav. 32, kur linijų nerado naudojant „Hough“ transformaciją. Tikimybinė „Hough“ transformacija linijų rado, tačiau nuotraukoje matoma, kad visos rastos linijos yra vertikalios.

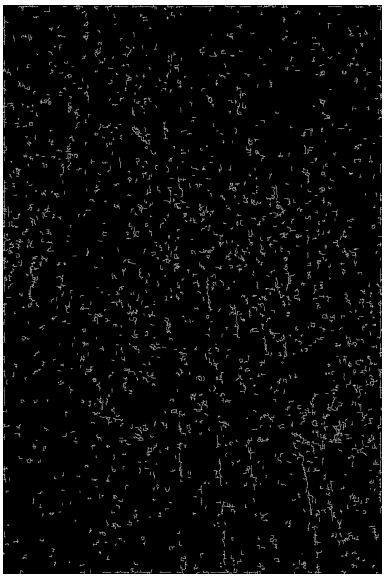
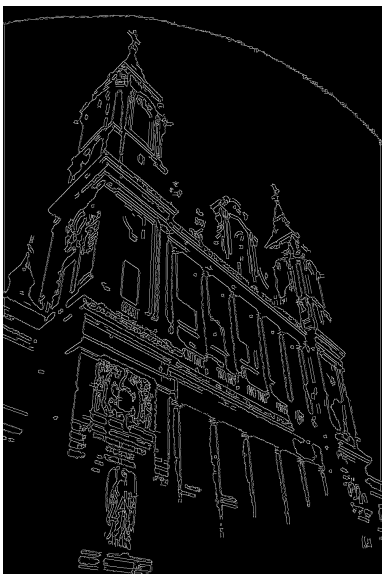
Iš 6 ir 7 lentelių galima daryti išvadą, jog kai analizuojama savaime „triukšminga“ nuotrauka, t.y. tokia, kuriai nors viena linijų uždėjimo funkcija nesuveikia su pirmu testu, normaliojo pasiskirstymo metu sugeneruoto triukšmo paveiktai nuotraukai geriausiai linijas pavyko uždėti ketvirto testo metu, tuo tarpu nuotrauka su DIP pirmo sprendimo triukšmu testų nepraėjo, o DIP antras sprendimas didelio poveikio linijų uždėjimui nepadarė.

6 lentelė. Testų parametrų rezultatai su nuotrauka Nr. 5., remiantis linijų radimo parametrų rezultatais. Kraštų aptikimo algoritmas - „Canny“, gauti paveiksliukai prieduose pav. 12 ir 13.

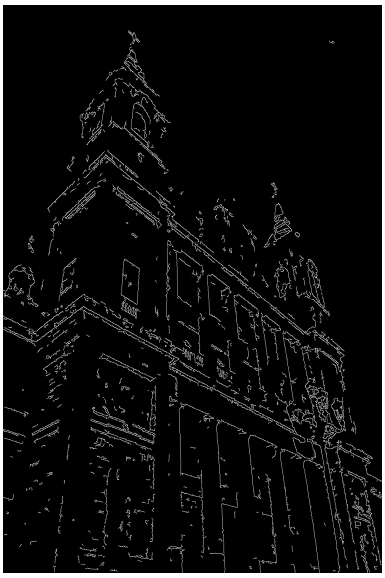
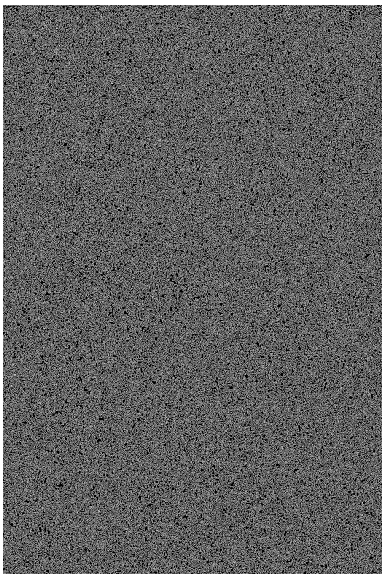
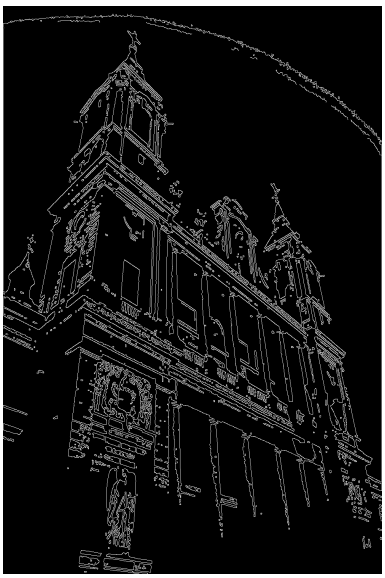
	„Hough“			Tik. „Hough“			
Testo nr. ir triukšmo tipas	Linijų sk.	Riba	Kampas (radianais)	Linijų sk.	Riba	Min. linijos ilgis	Maks. tarpas
1.	0	250	✗	3	250	250	40
2. NS	0	250	✗	0	✗	✗	✗
2. DIP 1 spr.	0	250	✗	2	250	250	40
2. DIP 2 spr.	0	250	✗	9	250	250	40
3. NS	0	250	✗	0	✗	✗	✗
3. DIP 1 spr.	0	250	✗	0	✗	✗	✗
3. DIP 2 spr.	0	250	✗	3	250	250	40
4. NS	250	7	1.5358897 - 1.553343	2	250	250	40
4. DIP 1 spr.	0	250	✗	0	✗	✗	✗
4. DIP 2 spr.	250	5	1.5358897 - 1.553343	9	250	250	40
5. NS	0	250	✗	0	✗	✗	✗
5. DIP 1 spr.	0	250	✗	0	✗	✗	✗
5. DIP 2 spr.	250	2	1.5358897	7	250	250	40

7 lentelė. Testų rezultatai su nuotrauka Nr. 5., remiantis, ar linijos rastos iš lentelės 6 duomenų. Kraštų aptikimo algoritmas - „Canny“.

Testo numeris	„Hough“			Tik. „Hough“		
1	✗			✓		
	Norm. skir.	DIP 1 spr.	DIP 2 spr.	Norm. skir.	DIP 1 spr.	DIP 2 spr.
2	✗	✗	✗	✗	✓	✓
3	✗	✗	✓	✗	✗	✓
4	✓	✗	✓	✓	✗	✓
5 (neoficialus)	✗	✗	✓	✗	✗	✓

„Canny“. Normalus skirst.	„Hough“	Tikimybinė „Hough“
	Linijų nerado!	Linijų nerado!
„Canny“. DIP pirmas spr.	„Hough“	Tikimybinė „Hough“
	Linijų nerado!	Linijų nerado!
„Canny“. DIP antras spr.	„Hough“	Tikimybinė „Hough“
		

24 pav. Ketvirto testo rezultatai - linijų radimas naudojant „Canny“, triukšmo tipas: normalus skirstinys, DIP pirmas ir antras sprendimai. Nuotrauka Nr. 1.

„Canny“. Normalus skirst.	„Hough“	Tikimybinė „Hough“
		Linijų nerado!
„Canny“. DIP pirmas spr.	„Hough“	Tikimybinė „Hough“
	Linijų nerado!	Linijų nerado!
„Canny“. DIP antras spr.	„Hough“	Tikimybinė „Hough“
		

25 pav. Penkto (neoficialaus) testo rezultatai - linijų radimas naudojant „Canny“, triukšmo tipas: normalus skirstinys, DIP pirmas ir antras sprendimai. Nuotrauka Nr. 1.

3.5. Tyrimo išvados

Padarytos išvados:

- Linijų paieškos uždaviniui atlikti, geriausius rezultatus pademonstravo „Canny“ algoritmas (žr. skyrelį 3.1). Tuo tarpu po kraštų aptikimo algoritmų „Sobel“ ir „Scharr“, rastos linijos buvo įstrižos (pasvirusios 45 laipsniu kampu), ir patobulintos linijų radimo ir uždėjimo funkcijos (žr. skyrelį 2.4) tokias linijas atmetė bei grąžino rezultatą, jog horizontalių linijų rasti nepavyko.
- Normaliojo skirstinio, kai $\mu = 0$, o $\sigma = 60$ ir DIP metodo pirmo sprendimo, kai $prob = 0.3$ sugeneruotas triukšmas visiškai sugadina nuotrauką, todėl linijų nebuvo rasta (žr. skyrelį 3.3).
- DIP antro sprendimo būdu sugenetuotas triukšmas, kai $\sigma = 0.33$ ir $\sigma = 0.005$, įtakos kraštų ir linijų paieškai nepadarė (žr. skyrelius 3.2 ir 3.3).
- Nuotraukai su pridėtiniu normaliojo skirstinio triukšmu, kai $\mu = 0$, o $\sigma = 60$, pritaikius Gauso filtrą su didžiausią suliejimą padariusiais parametrais iš pav. 3, kuomet kernelio dydis 7, o $\sigma = 60$, linijų neranda, reikia mažinti Gauso filtro parametrus (žr. skyrelį 3.4).
- Savaime „triukšmingai“ nuotraukai, kuomet jai nerandama linijų net nepridėjus triukšmo, geriausiai tinka naudoti Gauso filtrą, kuomet kernelio dydis 7, o $\sigma = 60$, kai pridėtinis triukšmas yra sugeneruotas normaliojo skirstinio būdo (žr. skyrelį 3.4).

Dėl šių išvadų, toliau tiesinimo uždaviniui įgyvendinti bus naudojamas „Canny“ algoritmas kraštų radimui ir paprasta „Hough“ transformacija. Gauso filtro parametrai savaime triukšmingai nuotraukai bus parinkti $K = 5$ arba 7, o $\sigma = 0$ arba 30 (30 pasirinkimas vietoj 60, nes 60 netinka savaime „netriukšmingoms“ nuotraukoms, o vėliau pridėtinis triukšmas nebus naudojamas), priklausmai nuo poreikio, apie tai plačiau skyrelyje 5.1.

4. Duomenų parinkimas tiesinimo uždaviniui

Po kraštų ir linijų radimo tyrimo, buvo pastebėta, kokios nuotraukos tinkamiausios tiesinimo uždaviniui atlikti. Tiesinimo uždaviniui testuoti, buvo rinktos nuotraukos, kurios turi pastebimą horizontą atspindinčias linijas, pagal kurias būtų galima remtis tiesinant nuotraukas rankiniu būdu naudojant bet kokią nuotraukų retušavimo programą. Taip pat, buvo atsižvelgta, kokios nuotraukos naudojamos RotNet modelio apmokymui.

Tiesinimo modelio ir neuroninio tinklo išbandymui buvo atrinkta 50 nuotraukų, kurios tokio tipo: gatvės vaizdai, saulėlydis, dominuojantys objektai (bet netinka fotografuoti iš šono), turintys horizontalią liniją - pastatai, turėklai, durys, jūros ir dangaus vaizdas ir t.t. 45-ios nuotraukos iš asmeninio archyvo ir 5-ios iš „Google Street View“ [15].

Originalios ištiesintos nuotraukos buvo pakreiptos rankiniu būdu naudojant nuotraukų redegavimo įrankį. Kreipimo kampas negalėjo būti daugiau nei 45 laipsniai, nes linijų radimo funkcija, kuri naudoja „Hough“ transformaciją aprašyta plačiau skyrelyje 2.4, atmeta linijas pakrypusias daugiau nei 45 laipsniai kampu. Kreipimo kampai, kurie reikalingi ištaisyti nuotraukas, sudėti į tiesinimo kampų masyvą. Šis panaudotas tiesinimo modelio ir RotNet modelių skaičiavimo metu, kur jeigu nuotraukos pasukimo kampas +-5 laipsniai skiriasi nuo rankiniu būdu pasuktų nuotraukų

kampo reikšmės, ir tik tuo atveju, jeigu sukama į tą pačią pusę (pvz.: jeigu TKK yra 3 laipsniai, bet bandoma sukti su -1, toks variantas nėra užskaitomas), tuomet bus teigiama, jog nuotrauka ištiesinta teisingai. Rankiniu būdu pasuktų nuotraukų kampas toliau bus minimas trumpiniu **TTK** - teisingas tiesinimo kampas.

Nuotraukos buvo sunumeruotos nuo 1 iki 50 (palengvinant tolesnio tyrimo aprašymą). Nuotraukos, kurių ID yra 7, 8, 9, 15 ir 22 yra iš „Google Street View“ [15] duomenų rinkinio. Kitos nuotraukos iš asmeninio archyvo padarytos su IPhone 6s, Nikon D3200 arba Nikon D5600 veidrodine kamera. Nuotraukų spalvų erdvė sRGB, dydis: mažiausias 402 KB, didžiausias 13 MB, dimensija: mažiausia 698 $\times$ 920 pikselių, didžiausia nuotraukos dimensija siekia 4000 $\times$ 6000 pikselių, horizontali / vertikali rezoliucija: mažiausia m96 DPI, didžiausia m300 DPI.

5. Nuotraukų tiesinimo modelis

Po atlikto tyrimo gautų išvadų su kraštų aptikimo ir linijų radimo algoritmais, toliau tiesinimo uždaviniui įgyvendinti bus naudojamas „Canny“ algoritmas kraštų radimui ir paprasta „Hough“ transformacija. Buvo sudarytas tiesinimo modelis, kuris dokumente toliau taip ir vadinamas.

5.1. Modelio aprašymas

Buvo įgyvendintas modelis tiesinimo uždaviniui išspręsti. Modelio vizualizacija pateikta pav. 26. Modelio buvo panaudotos anksčiau suprogramuotos funkcijos, ieškančios kraštų ir linijų, daugiau apie jas skyrelyje 7, kur taip pat pateiktas detalesnis indėlio į tiesinimo modelio programavimą aprašymas. Detali modelio eiga:

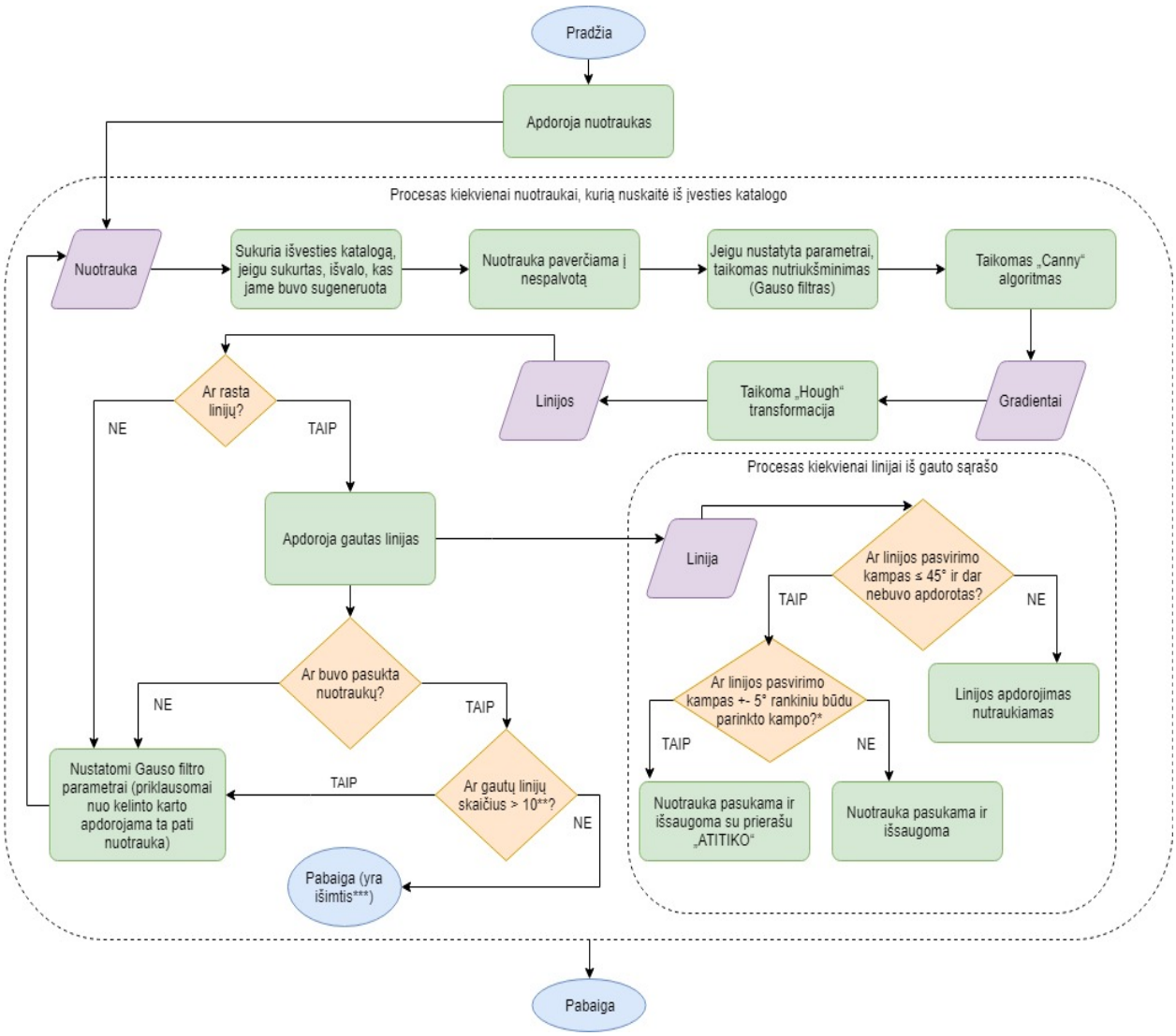
- Iš įvesties katalogo nuskaitomos nuotraukos, kurias norima ištiesinti. Pradedama nuotraukų iteracija.
- Sukuriamas išvesties katalogas, kuriame bus išsaugomos nuotraukos: saugoma gradientų nuotrauka (kraštai) po „Canny“ algoritmo, nuotrauka su uždėtomis linijomis (šios nuotraukos buvo skirtos validacijai, parametrų pagalba, galima atsisakyti šių nuotraukų saugojimo), pasuktos nuotraukos. Maksimaliai pasuktų nuotraukų (**MPN** - kode vadinamas *picturesToRotate*) skaičius pagal numatymą skaičius parinktas 10.
- Nuotrauka paverčiama į nespalvotą. Šis veiksmas būtinas prieš „Canny“ algoritmo leidimą.
- Jeigu nustatyti Gauso filtro parametrai (kernelio dydis ne 0), tuomet nuotraukai pritaikomas Gauso filtras (nutriukšminimas).
- Taikomas „Canny“ algoritmas gradientų radimui.
- Kraštai (gradientai) paduodami „Hough“ tranformacijai, ieškomos linijos. Jeigu linijų buvo rasta, vyksta iteracija per jas, kurios metu:
  - Apdorojama viena linija. Jeigu linijos pokrypio kampas tarp 0 ir 45 laipsnių ir šis kampas nebuvo aprotas ankstesnėse iteracijose, linija tinka ir imama jos posūkio kampo reikšmė.

  - Jeigu posūkio kampo reikšmė +- 5 laipsniais skiriasi nuo TTK kampo reikšmės ir jeigu sukama į tą pačią pusę (pliusinis kampas rodo, kad sukama į dešinę, minusinė kampo reikšmė, jog nuotrauka sukama į kairę pusę), prie ištiesintos nuotraukos pavadinimo bus pridėtas prierašas „ATITIKO“. Šis veiksmas vykdomas tik tuo atveju, jeigu rankiniu būdu pasuktų nuotraukų kampų masyvas yra nurodytas.
  - Nuotrauka pakreipiama ir apkerpama, išsaugota su nauju pavadinimu (originalus pavadinimas + prierašas (jeigu yra) + indeksas).

- Vykdomas pakartotinis procesas, kai taikomas Gauso filtras, jeigu įvyksta nors vienas iš žemiau pateiktų atvejų:
  - Jeigu pasukta nors viena nuotrauka, tačiau rastų linijų skaičius viršija nustatytą norimą pasuktų nuotraukų skaičių.
  - Jeigu nebuvo pasukta nuotraukų.
  - Jeigu nerado linijų.

- Gauso filtro iteracijos buvo nustatytos keturios (iš prieš tai vykdyto tyrimo gautų rezultatų): kai $K = 5$ ir $\sigma = 0$, kai $K = 5$ ir $\sigma = 30$, kai $K = 7$ ir $\sigma = 0$, kai $K = 7$ ir $\sigma = 30$.

- Išimties atvejis: jeigu linijų radimo algoritmas „Hough“ grąžino linijų daugiau nei norima pasukti nuotraukų nustatytas skaičius ir nuotraukų buvo pasukta, vykdomos Gauso filtro iteracijos tol, kol atitinka kitus kriterijus. Tačiau buvo pastebėta, kad kartais pritaikius gauso filtrą, su didesniais parametrais, linijų nebuvo rasta. Todėl vyksta atgalinis procesas, kur randama, kuriuo metu buvo pasukta nuotraukų, netgi tada, kai rastų linijų skaičius didesnis už nustatytą norimą pasuktų nuotraukų skaičių.



26 pav. Nuotraukos tiesinimo modelis naudojant „Canny“ kraštų aptikimo ir „Hough“ linijų radimo algoritmus. Žvaigždučių paaiškinimas: * „ATITIKO“ prierašas pridedamas tik testavimo atveju, kai nurodytas rankiniu būdu pasuktų nuotraukų kampų masyvas. ** Šis skaičius priklauso nuo skaičiaus, kiek maksimaliai pasuktų nuotraukų norima gauti, parametras gali būti koreguojamas. *** Išimtis - jeigu po Gauso filtro pritaikymo buvo nerasta linijų iš viso, bet po praeito proceso buvo pasukta nuotraukų, leidžiamas praeitas procesas kaip galutinis nuotraukos apdorojimo veiksmas.

5.2. Tiesinimo modelio tyrimas

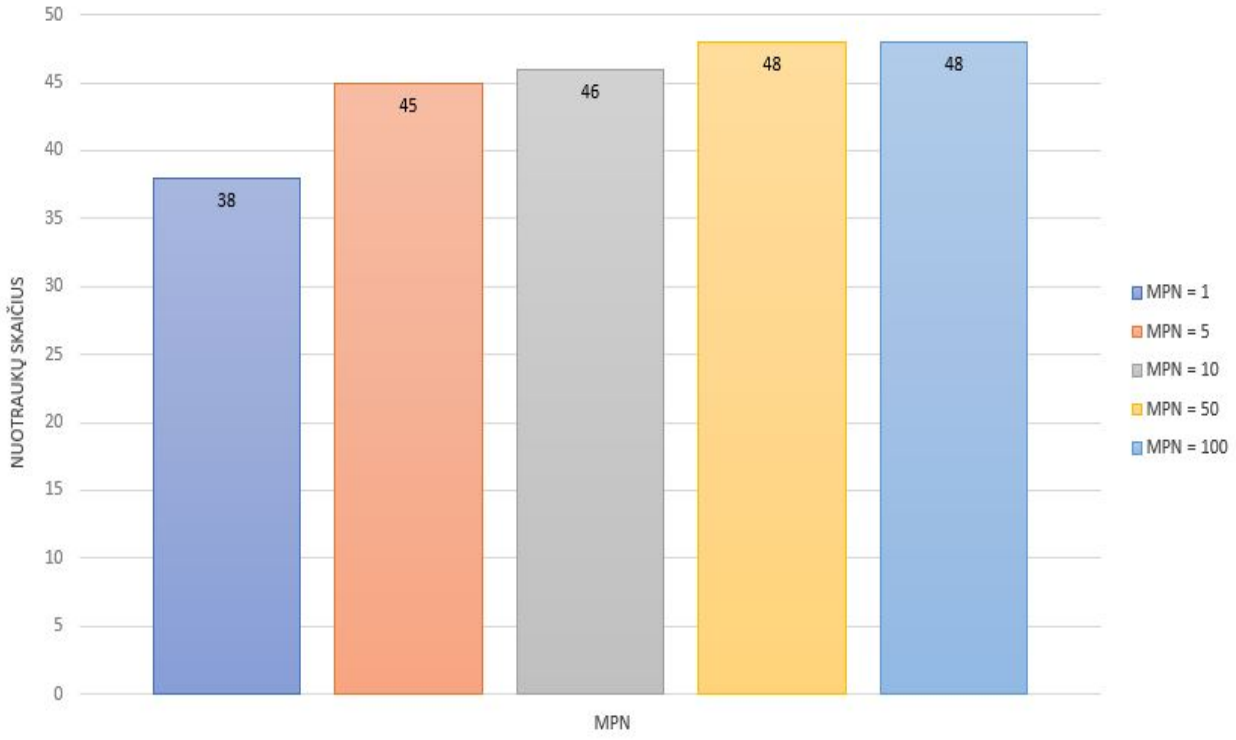
Tiesinimo modelis buvo bandytas penkis kartus, kuomet keitėsi maksimalus pasuktų nuotraukų skaičius (MPN) - parinktos reikšmės yra 1, 5, 10, 50, 100. Kodėl šis parametras MPN svarbus sukurtas tiesinimo medeliui? Šis skaičius buvo įvestas, nes nuotraukoje randama daugiau linijų negu viena, ir pirmoji linija ne visada yra ta, kuri yra ideali tiesinimui - „Hough“ transformacijos metu grąžintos linijos sąraše pateikiamos atsitiktinai. Todėl pasiūlytas variantas yra tiesinti daugiau daugiau nuotraukų nei vieną ir vėliau išsirinkti tinkamiausią.

Kiekvieno bandymo metu, jeigu su testuojama nuotrauka gautas nors vienas pasuktas variantas su prierašu „ATITIKO“, laikoma, jog **nuotrauką pavyko ištiesinti**. Pavyzdys, kur pateikti gauti skirtumai, tarp tiesinimo kampų ir TTK yra lentelėse 14 ir 15, kai tiesinimo modelio parametras

MPN lygus 10.

Bandymų rezultatai pateikti pav. 27. Iš gautos diagramos matoma, kad nustačius, jog tiesinimo modelis ištiesintų remiantis pirma gauta linija ir pateiktų tik viena ištiesintą variantą, buvo ištiesintos 37-ios testavimo nuotraukos, kai tiesinimo kampas nenukrypęs daugiau nei 5 laipsniai nuo testavimo nuotraukos TTK reikšmės. Taip pat, diagrama parodo, jog MPN nustačius 5 ir 10 ištiesintos atitinkamai 45 ir 46 nuotraukos, kai MPN reikšmė buvo lygi 10, pavyko ištiesinti nuotrauką, kurios ID lygus 38.

Daugiausia testavimo nuotraukų ištiesinta, kai MPN nustatyta 50 ir 100, tačiau šiuo atveju, kai kurios nuotraukos turėjo iki 50/100 pasuktų (tiesinimo modelio pasiūlytų) nuotraukų: iš lentelės 9 duomenų, daugiausia pasiūlytų tiesinimo variantų, viso 67, turėjo nuotrauka, kurios ID lygus 46. Toliau lentelėse 8 ir 9 bus analizuojami duomenys iš bandymo su MPN reikšme 100. Kai MPN lygus 50, tik nuotrauka, kurios ID 46, turėjo mažiau sugeneruotų ištiesintų nuotraukų nei tada, kad MPN lygus 100, taigi abiejų bandymų atsakymai beveik tokie patys.

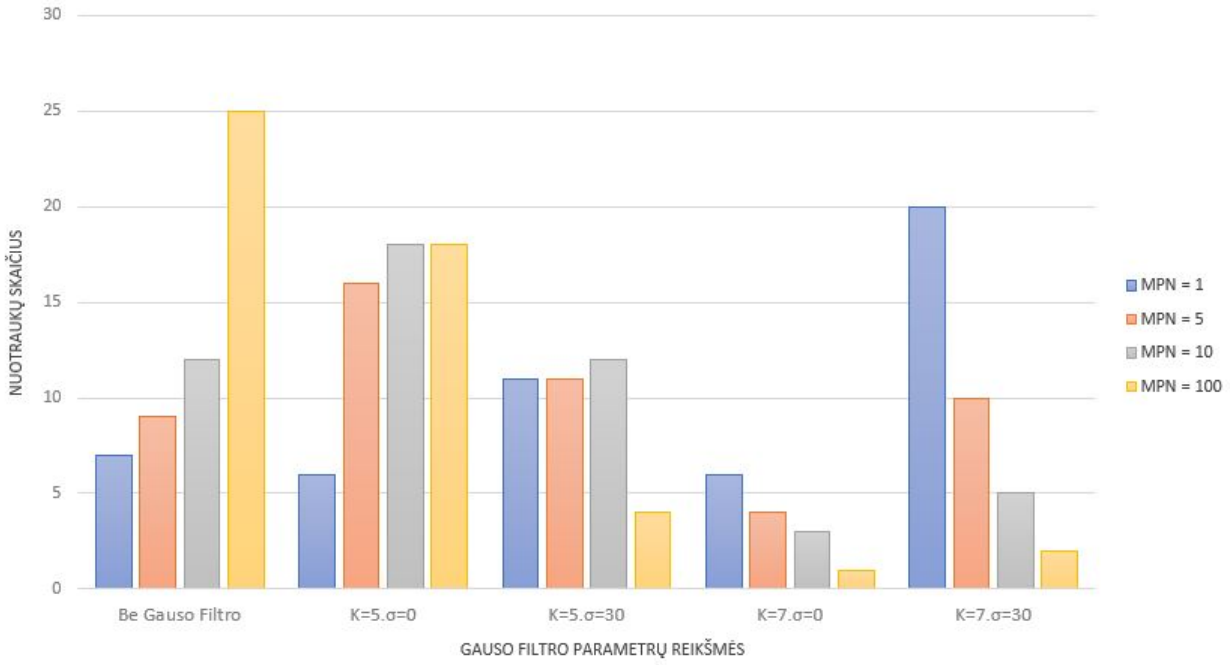


27 pav. Nuotraukos tiesinimo modelio bandymo rezultatai, keičiant MPN reikšmę - parinktos reikšmės yra 1, 5, 10, 50, 100. Testavimo nuotrauka laikoma ištiesinta, jeigu su testuojama nuotrauka gautas nors vienas pasuktas variantas su prierašu „ATITIKO“.

Iš diagramos pav. 28 matoma, kad kuo mažesnė MPN reikšmė, tuo modelis taikė didesnius Gauso filtro parametrus. Taip įvyko dėl to, jog su mažesne MPN reikšme modelis stengiasi išanalizuoti tik kuo svarbesnes linijas (tikrinama, ar gautų linijų skaičius daugiau už MPN, jeigu taip, taiko didesnius Gauso filtro parametrus, tam, kad sumažintų gautų linijų kiekį). Tačiau tokiu atveju, galima pašalinti horizontą atspindinčias linijas, todėl derėtų rinktis didesnę nei 1 MPN parametro reiškmę.

Po gautų bandymo rezultatų, pasiūlymas būtų rinktis MPN reikšmę lygią 5 - šiuo atveju pavyko ištiesinti 90% testavimo nuotraukų ir pasiūlyta mažiau ištiesintų nuotraukų variantų, negu tuomet, kai MPN reikšmė lygi 10 arba 100 (žr. lenteles 8 ir 9, kur nurodytas pasuktų nuotraukų skaičius kiekvienai testavimo nuotraukai, pvz.: MPN pasiektas su nuotraukomis, kurių ID yra 37, 38, 39,

44, 46. Reiktų peržiūrėti visus dešimt pasiūlytų tiesinimo variantų ir atsirinkti tinkamiausią). Kai MPN lygus 10, modelis sugeneravo 168 ištiesintas nuotraukas - 56 nuotraukom daugiau, nei tada, kai MPN lygus 5, o pavykusių ištiesintų nuotraukų skaičius skiriasi tik per vieną nuotrauką. Kai MPN lygu 100 buvo ištiesinta 96% testavimo nuotraukų, tačiau reiktų peržiūrėti 435 testavimo modelio sugeneruotus tiesinimo variantus.



28 pav. Gauso Filtro parametrų panaudojimo diagrama, keičiant MPN reikšmę.

8 lentelė. Tiesinimo modelio bandymo rezultatai, nuotraukų ID nuo 1 iki 30. PNS - pasuktų nuotraukų skaičius, GFP - Gauso filtro parametrų reikšmė, brūkšnys žymi, kad Gauso filtras nebuvo naudotas. Raudonai pažymėtos PNS, tarp kurių nebuvo nė vieno ištiesinto varianto.

	MPN = 1		MPN = 5		MPN = 10		MPN = 100	
ID	PNS	GFP	PNS	GFP	PNS	GFP	PNS	GFP
1	1	-	1	-	1	-	1	-
2	1	K=5. σ=30	2	K=5. σ=0	2	K=5. σ=0	4	-
3	1	K=7. σ=30	3	K=5. σ=30	3	K=5. σ=30	19	K=5. σ=0
4	1	-	1	-	1	-	1	-
5	1	K=7. σ=30	1	K=7. σ=30	7	K=5. σ=30	7	K=5. σ=30
6	1	K=7. σ=30	2	K=7. σ=0	2	K=7. σ=0	5	-
7	1	K=7. σ=30	4	K=5. σ=0	4	K=5. σ=0	10	-
8	1	K=5. σ=30	5	K=5. σ=0	5	K=5. σ=0	5	K=5. σ=0
9	1	K=7. σ=30	4	K=5. σ=30	7	K=5. σ=0	25	-
10	1	K=7. σ=30	3	K=7. σ=0	3	K=5. σ=30	7	-
11	1	K=7. σ=30	3	K=5. σ=30	3	K=5. σ=30	7	-
12	1	-	2	-	2	-	2	-
13	1	K=5. σ=0	1	K=5. σ=0	1	K=5. σ=0	1	K=5. σ=0
14	1	K=7. σ=30	1	K=5. σ=30	3	K=5. σ=0	13	-
15	1	K=7. σ=0	2	K=5. σ=30	4	K=5. σ=0	8	-
16	1	K=5. σ=30	2	K=5. σ=0	2	K=5. σ=0	19	-
17	1	K=7. σ=30	2	K=7. σ=30	2	K=7. σ=30	3	K=5. σ=30
18	1	K=7. σ=30	1	K=5. σ=30	1	K=5. σ=30	19	K=5. σ=0
19	1	K=5. σ=30	1	K=5. σ=30	1	K=5. σ=30	5	K=5. σ=0
20	1	K=5. σ=30	2	K=7. σ=30	2	K=5. σ=30	2	K=5. σ=30
21	1	K=5. σ=30	1	K=5. σ=0	1	K=5. σ=0	4	-
22	1	K=7. σ=0	2	K=5. σ=0	2	K=5. σ=0	2	K=5. σ=0
23	1	K=7. σ=0	1	K=7. σ=0	1	K=7. σ=0	1	K=7. σ=0
24	1	-	2	-	2	-	2	-
25	1	K=7. σ=0	1	K=5. σ=0	2	-	2	-
26	1	K=7. σ=30	1	K=7. σ=30	1	K=7. σ=30	12	-
27	1	K=5. σ=30	2	K=5. σ=0	2	K=5. σ=0	2	K=5. σ=0
28	1	K=7. σ=30	1	K=7. σ=30	1	K=5. σ=30	10	K=5. σ=0
29	1	K=7. σ=30	1	K=7. σ=30	1	K=5. σ=30	5	-

9 lentelė. Tiesinimo modelio bandymo rezultatai, nuotraukų ID nuo 30 iki 50. PNS - pasuktų nuotraukų skaičius, GFP - Gauso filtro parametrų reikšmė, brūkšnys žymi, kad Gauso filtras nebuvo naudotas. Raudonai pažymėtos PNS, tarp kurių nebuvo nė vieno ištiesinto varianto. Gale gautos sumos, kiek viso kiekvienu bandymu buvo sugeneruota tiesinti bandytų nuotraukų.

	MPN = 1		MPN = 5		MPN = 10		MPN = 100	
ID	PNS	GFP	PNS	GFP	PNS	GFP	PNS	GFP
30	1	K=7. σ=0	2	K=5. σ=30	6	K=5. σ=0	6	K=5. σ=0
31	1	K=7. σ=0	2	K=7. σ=30	2	K=7. σ=30	18	-
32	1	K=5. σ=30	1	K=5. σ=30	1	K=5. σ=30	1	K=5. σ=30
33	1	K=5. σ=0	2	-	2	-	2	-
34	1	K=7. σ=30	2	K=7. σ=30	2	K=7. σ=30	12	K=5. σ=0
35	1	K=7. σ=30	3	K=5. σ=30	3	K=5. σ=30	8	K=5. σ=0
36	1	K=7. σ=30	2	K=5. σ=0	2	K=5. σ=0	2	K=5. σ=0
37	1	-	5	-	10	-	18	-
38	1	K=7. σ=30	5	K=7. σ=30	10	K=7. σ=0	11	K=7. σ=30
39	1	K=5. σ=0	5	K=5. σ=0	10	K=5. σ=0	18	K=5. σ=0
40	1	K=7. σ=30	2	K=5. σ=30	7	K=5. σ=0	7	K=5. σ=0
41	1	K=5. σ=30	2	K=5. σ=0	3	-	3	-
42	1	K=5. σ=30	3	K=5. σ=0	3	K=5. σ=0	3	K=5. σ=0
43	1	K=5. σ=30	3	K=5. σ=0	3	K=5. σ=0	3	K=5. σ=0
44	1	-	5	-	10	-	32	-
45	1	K=5. σ=0	1	K=5. σ=0	1	K=5. σ=0	1	K=5. σ=0
46	1	K=7. σ=30	5	K=7. σ=30	10	K=7. σ=30	67	K=7. σ=30
47	1	-	3	-	3	-	3	-
48	1	K=5. σ=0	1	K=5. σ=0	6	-	6	-
49	1	K=7. σ=30	2	K=7. σ=0	4	K=5. σ=30	10	K=5. σ=0
50	1	K=5. σ=0	1	-	1	-	1	-
$\sum$	50		112		168		435	

6. RotNet neuroninio tinklo tyrimas

Tyrimui su neuroniniu tinklu atlikti, buvo pasirinktas RotNet modelis, kurio panaudojimas detaliai aprašytas straipsnyje „Correcting Image Orientation Using Convolutional Neural Networks“ [29]. Modelio kodas paimtas iš *GitHub* talpyklos. Šioje talpykloje yra kodas, reikalingas apmokyti ir išbandyti konvoliucinį neuroninį tinklą (CNN, naudotas ResNet50 algoritmas), kad būtų galima numatyti nuotraukos pasukimo kampą, tam kad būtų ištaisyta jo padėtis [6].

6.1. Apmokymo aplinka

Paleisti RotNet modelį buvo pakurta virtuali mašina „Microsoft Azure“ debesyje. Mašinos duomenys: operacinė sistema - Linux Ubuntu 18.04-LTS, dydis - *Standard NC6 Promo* (6 vcpus, 56 GiB atminties), lokacija - šiaurės Europa, diskas - 128 GiB (*Standard SSD*).

Buvo sudiegti „GPU drivers“ naudojantis oficialia „Microsoft Azure“ dokumentacija [7], užtruko rasti klaidą, jog papildomas bibliotekų kelias turi būti eksportuojamas:

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda-10.2/targets/x86_64-linux/lib

Po to buvo suintaliuotos bibliotekos reikalingos RotNet modeliui, jų sąrašas pateiktas talpykloje su pavadinimu „requirements.txt“ [6].

6.2. Duomenų rinkiniai

Buvo parinkti trys duomenų rinkiniai tinklui apmokyti.

Pirmasis pasiūlytas ir naudotas RotNet modelio duomenų rinkinys yra „Google Street View“ nuotraukos [15]. Šį rinkinį sudaro apie 62058 nuotraukų. Tinklas nustojo tobulėti po 35 epochų, kampo reikšmės paklaida $3,0751$. Gautas apmokytas modelis pavadintas *rotnet_street_view_resnet50.hdf5*, toliau dokumente vadinsis **„Street View Model"** arba „Street View“ modelis.

Antrasis duomenų rinkinys yra paimtas iš *Kaggle* duomenų saugyklos [33]. Duomenų rinkinį sudaro 82783 įvaraus tipo nuotraukų - žmonės, gatvės vaizdai, maistas, pastatai ir t.t. Tinklas nustojo tobulėti po 36 epochų, kampo reiškmės paklaida $16,2256$. Gautas apmokytas modelis pavadintas $rotnet\_resnet50\_Ep50RandomPictures.hdf5$, toliau dokumente vadinsis **„Random Pictures Model"** arba „Random Pictures" modelis.

Trečiąjį duomenų rinkinį sudaro pastatų nuotraukos „Pasadena Houses“ [38] ir „Google Street View“ $part1$ ir $part2$ [15]. Rinkiniui sudaryti buvo panaudota dalis kodo, parašyto tiesinimo medeliui 5, nuotraukos buvo patiesintos, jeigu rasta linija yra pasvirusi +-5 laipsniais. Gautos nuotraukos, kurios atitinka tokią taisyklę, išsaugojamos kataloge, vėliau naudotam RotNet modeliui apmokyti, jeigu nuotrauka neatitinka kriterijaus, išsaugoma originali versija. Gautą duomenų rinkinį sudaro 10340 nuotraukos. Tinklas nustojo tobulėti po 31 epochos, kampo reiškmės paklaida $7,0427$. Gautas apmokytas modelis pavadintas $rotnet\_resnet50\_Ep50Archive.hdf5$, toliau dokumente vadinsis **„Archives Model"** arba „Archives" modelis.

Tinklas buvo apmokytas naudojant visus tris duomenų rinkinius, viso su 155181 nuotraukų. Tinklas nustojo tobulėti po 43 epochų, kampo reikšmės paklaida $10,7084$. Gautas apmokytas modelis pavadintas $rotnet\_all\_resnet50.hdf5$, toliau dokumente vadinsis **„All Pictures Model"** arba „All Pictures" modelis.

Kiekvieno apmokymo metu 90% nuotraukų iš rinkinio buvo skirta apmokymui, 10% nuotraukų testavimui.

6.3. Tyrimas ir palyginimas su tiesinimo modeliu

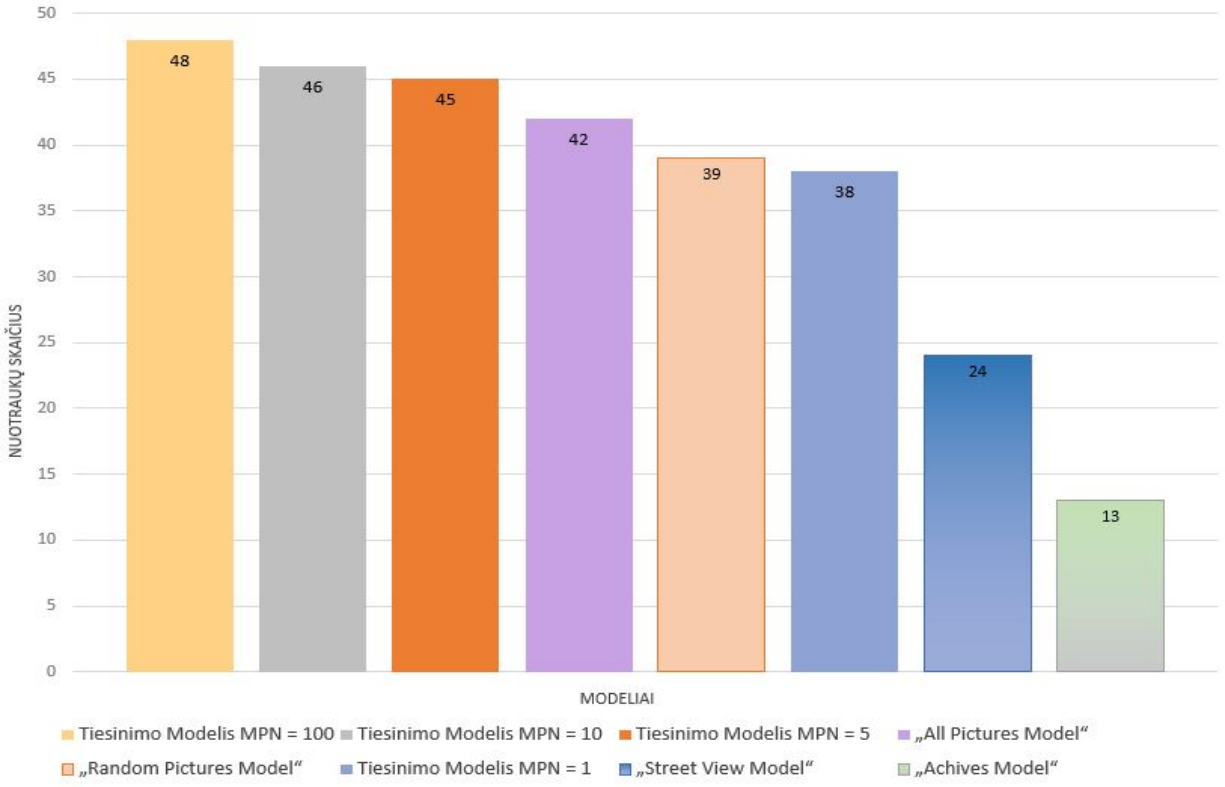
Prieduose esančiose lentelėse 16 ir 17 pavaizduota gautų kampų reikšmės, kuriomis buvo pasuktos nuotraukos kiekvieno testavime dalyvausio modelio (tiesinimo modelio paimtos nuotraukos, kai parametro MPN reikšmė buvo lygi 10).

29 pav. Atrinktos iškreiptos nuotraukos ID 24, 26, 27, 28, 32 (eilutės numeris 1, virš nuotraukos TKK reikšmė) lyginamos su tiesinimo modelio, kai MPN lygu 10, gautais rezultatais (eilutės numeris 2) bei su RotNet modelių gautomis nuotraukomis: „Street View Model" (eilutės numeris 3), „Random Pictures Model" (eilutės numeris 4), „Archives Model“ (eilutės numeris 5) ir „All Pictures Model“ (eilutės numeris 6). Virš nuotraukos nurodytas nuotraukos numeris ir modelio parinkto tiesinimui laipsnio reikšmė.

Aukščiau pav. 29 pateikti gauti tiesinimo rezultatai nuotraukų ID 24, 26, 27, 28 ir 32 gauti

tiesinant RotNet modelius ir tiesinimo modelį, kai MPN lygu 10, aprašytą detaliau skyrelyje 5. Šis pavyzdys yra svarbus, nes čia pavaizduotos dvi nuotraukos, kurių nepavyko ištiesinti tiesinimo modeliui, kai MPN buvo lygus 50 ir 100, šių nuotraukų ID yra 24 ir 32. Tačiau abi šias nuotraukas pavyko ištiesinti su RotNet modeliu „Random Pictures Model", kurį apmokyti buvo naudojamas didžiausias duomenų rinkinys iš visų turimų trijų. Nuotrauką, kurios ID lygus 32, pavyko ištiesinti ir su RotNet „Street View Model", mat nuotraukoje egzistuoja kelio vaizdas. Kodėl tiesinimo modeliui nepavyko susidoroti su šiomis dvejomis nuotraukomis? Todėl, nes sprendžiant iš rastų linijų, šiose nuotraukose dominuoja vertikalios linijos (pasvirę daugiau nei 45 laipsniai kampu), o tokios linijos modelyje yra atmetamos. Kiti ištiesintų nuotraukų pavyzdžiai pateikti prieduose pav. 33, 34 ir 35.

Ištiesinta nuotrauka naudojant bet kurį iš trijų RotNet modelių laikoma ištiesinta, jeigu sukimo kampas yra +-5 laipsniai nuo TTK reikšmės ir tik tada, jeigu sukimo kampas į tą pačią pusę, daugiau apie tai skyrelyje 4. RotNet modeliai grąžina tik po vieną tiesinimo variantą (lygiai taip pat, kaip tiesinimo modelis, kai MPN lygu 1). Gauti rezultatai susumuoti ir pavaizduoti diagramoje 30. Iš diagramos matoma, jog geriausius rezultatus pademonstravo RotNet „All Pictures Model“, kuriam apmokyti buvo naudojami visi trys duomenų rinkiniai – šiam modeliui pavyko ištiesinti 86% testavimui naudotų nuotraukų. RotNet „Random Pictures Model“ modeliui pavyko ištiesinti 39 nuotraukas iš 50 testavimui skirtų nuotraukų (78%). Kiti du RotNet modeliai pademosntravo žemus rezultatus: „Street view Model“ ištiesino šiek tiek mažiau nei pusę nuotraukų (ištiesintos nuotraukos yra gatvės vaizdai), „Archives Model“ ištiesino tik 26% testui skirtų nuotraukų.



30 pav. Pasuktų nuotraukų skaičius, kai skirtumas tarp TTK +-5 laipsniai ir jeigu nuotrauka sukama į tą pačią pusę, kiekvienam testavime dalyvavusiam tiesinimo modeliui.

Tiesinimo modelio pliusai: nustačius didesnę MPN parametro reikšmę, galima gauti daugiau ištiesintų nuotraukų, tačiau tokiu atveju padidėja sugeneruotų nuotraukų skaičius, nes modelis pateikia daugiau ištiesintų egzempliorių, t.y. analizuoja daugiau linijų. Dar vienas modelio minusas,

tas kad analizuoti sugeba tik nuotraukas, kurios pakreiptos iki 45 laipsnių kampu.

RotNet modelių pliusai: gali analizuoti nuotraukas pakreiptas bet kokiu kampu, visada grąžins tik vieną ištiesinimo variantą. Minusai: reikia gana didelio ir įvairaus duomenų rinkinio apmokyti tinklą. Tinklo mokymas trunka ilgai ir reikia paruošti apmokymui skirtą aplinką.

RotNet modeliai tiktų tiesinimo uždaviniui išspręsti geriau, jeigu jiems apmokyti būtų parinkti saviti, būtingi tik tam tikrai nuotraukų grupei, ir gausesni nuotraukų duomenų rinkiniai, pvz.: RotNet „Street View Model“ tinka tiesinti miesto vaizdų, kelių nuotraukas, tačiau nesusidorojo su kitokio tipo nuotraukom, pvz.: saulėlydžio, kai kurių pastatų (bokštai) vaizdais.

7. Programavimo dalies aprašymas

Tyrimui atlikti, kodas buvo programuojamas „Python“ (versija 3.7.1) programavimo kalba, „OpenCV“ biblioteka (versija 4.2.0), „Numpy“ biblioteka (versija 1.15.4), operacinė sistema - „Windows 10 Enterprise“, procesorius - AMD Ryzen 7 PRO 2700U x/ Radeon Vega Mobile Gfx 2.20 GHz, atmintis - 16GB, sistemos tipas - 64-bit.

Indėlis į kodo rašymą kraštų ir linijų algoritmų tyrimui atlikti:

- Parašytas bendras algoritmas kraštų radimui priklausomai nuo operatoriaus tipo: „Sobel“ ar „Scharr“. Plačiau poskyryje 2.3.
- Parašytos funkcijos triukšmui uždėti: normalio skirstinio, druskos ir pipirų (DIP) pirmas (pirmam atvejui idėja iš straipsnio *„Noise models in digital image processing“* [4]) ir antras atvejis. Plačiau poskyryje 2.2.
- Nustatytos kritinės reikšmės parametrams, įgyvendintas parametrų švelninimo žingsnis, tikimybinės „Hough“ transformacijos metu švelninami trys parametrai, tačiau paeiliui ir nustojama, kol visi pasiekia kritinę parametrų reikšmę. Tokia parametrų švelninimo realizacija padaryta funkcijose *linedPictureHoughLinesBestParametersChange* ir *linedPictureHoughLinesPBestParametersChange*, kurios suranda linijas, naudojant „Hough“ transformacijos ir tikimybinės „Hough“ transformacijos linijų radimo funkcijas iš „OpenCV“ bibliotekos [10], ir po to tas linijas uždeda ant originalios nuotraukos. Plačiau poskyryje 2.4.
- Patobulinta po „Hough“ transformacijos įvykdymo linijų uždėjimo dalis - tikrinamas gautas kampas θ, horizontalių linijų radimui šiame tyrime apibrėžiama, jog linijos bus įskaitytos jeigu θ reikšmė patenka tarp 0 ir 45 laipsnių. To paties padaryta nebuvo po tikimybinės „Hough“ transformacijos, nes ši funkcija negrąžina kampo reikšmės.
- Abiejų linijų radimo ir uždėjimo funkcijų metu, buvo realizuotas atmetimas linijų, kurios arčiau nei 5 pikseliai prie nuotraukos krašto (dėl DIP triukšmo tipo sugeneruoto triukšmo įtakos).

Indėlis į kodo rašymą programuojant tiesinimo modelį, naudojantį „Canny“ kraštų aptikimo ir „Hough“ linijų radimo algoritmus:

- Parašyta pagrindinė funkcija *DoMainRotation* tiesinimo veiksmų eigai atlikti: funkcija kviečia katalogo paruošimo/išvalymo funkciją, nuotrauką paverčia į nespalvotą, kviečia „Canny“ algoritmą, taiko Gauso filtrą, kviečia linijų uždėjimo ir tiesinimo funkciją *linedPictureHoughLinesBestParametersChangeRotate*. Ši funkcija prima parametrus: kelią iki origi-

nalios nuotraukos, kelią, kur bus saugomi rezultatai, Gauso filtrui skirtus parametrus (pagal numatymą, parametrai $K = 0$ ir $\sigma = 0$).

- Realizuota rezultatų katalogo paruošimo funkcija *removeRotatedImages*: sukuria katalogą, ir jeigu yra išsaugotų pasuktų nuotraukų, jas ištrina.
- Realizuota funkcija *linedPictureHoughLinesBestParametersChangeRotate*, kurios pagrindas paremtas anksčiau realizuotos funkcijos *linedPictureHoughLinesBestParametersChange* kodu. Patobulinimas: tikrinamos visos linijos (ne tik 40 pirmų), kai kurios linijos prasilenkia labai mažu skirtumu (pvz.: tūkstantosios dalies po kableliu), taigi, kampo reikšmė suapvalinta iki vienetų, ir tokios reikšmės kampas apdorojamas tik kartą. Funkcija tikrina su masyvu, kuriama užpildyta rankiniu būdu pasuktų nuotraukų kampai, tikrinama ar rastos linijos posūkio kampas +- 5 laipsniais skiriasi nuo rankiniu būdu pasuktos nuotraukos kampo ir ar kreipiama į tą pačią pusę (jeigu nuotraukos TTK yra 3, bet pagal rastą liniją reikia kreipti -2 laipsniais, tokia linija neužskaitoma). Jeigu taip, pridedamas prierašas. Ši funkcija ištiesina, apkarpo ir išsaugo pasuktą nuotrauką. Funkcija priima parametrus: kelią iki originalios nuotraukos, kelią iki išvesties katalogo, ribos reikšmę ir gradientus. Funkcija grąžina: nuotrauką su uždėtomis linijomis, $True$ / $False$ reikšmę, jeigu reikia bandyti tiesinimo metodą dar kartą panaudojus Gauso filtrą, $True$ / $False$ reikšmę, jeigu buvo ištiesinta nors viena nuotrauka.

Indėlis į kodo rašymą/pataisymai RotNet modelio, paimto iš *GitHub* talpyklos [6]:

- Patobulinimas: buvo pakeista bibliotekos pavadinimas *keras* į *tensorflow.keras*.
- Skripte $correct\_rotation.py$ pridėtas sukimo kampo spausdinimas bei 43-ioj eilutėj esantis parametras $val\_samples = len(image\_paths)$ pakeistas į $steps = len(image\_paths)$.
- Patobulintas paveikslėlių nuskaitymas per katalogus skripte $street\_view.py$ 29-33 eilutėse.

Atviro kodo (angl. *Open Source*) panaudojimas:

- Linijų paieškai naudotos „Hough“ transformacijos ir tikimybinės „Hough“ transformacijos funkcijos iš „OpenCV“ bibliotekos [10].
- Spalvota nuotrauka į nespalvotą buvo versta su „OpenCV“ funkcijos pagalba: *cv2.cvtColor(imageOriginal, cv2.COLOR_BGR2GRAY)* [8].
- Nuotraukų nuskaitymui ir įrašymui panaudotos „OpenCV“ bibliotekos funkcijos *cv2.imread()* ir *cv2.imwrite()* [12].
- Kitiems veiksmas atlikti, pvz.: sugeneruoti normaliojo skirstinio triukšmą, rasti medianą, ir t.t. buvo naudotos funkcijos iš „Numpy“ bibliotekos.
- „Canny“ kraštų algoritmas naudotas iš „OpenCV“ bibliotekos: *cv2.Canny(image, lower, upper)* [9], apatinės ir viršutinės ribų parinkimas buvo paremtas Adrian'o Rosebrock'io atliktu tyrimu [27].
- Gauso filtro funkcija naudota iš „OpenCV“ bibliotekos: *cv2.GaussianBlur(image, (K, K), sigma)* [11].
- Nuotraukos pasukimui ir apkirpimui naudotos funkcijos iš RotNet modelio *GitHub* talpyklos esančio skripto $utils.py$.

Išvados ir rekomendacijos

Šio magistro baigiamojo darbo **tikslas** buvo įgyvendintas — ištiesintos nuotraukos, naudojant dirbtinį neuroninį tinklą ir pasiūlytą tiesinimo metodą, kuriame panaudojami kraštų ir linijų aptikimo algoritmai. Šiam magistro baigiamajam darbui išsikelti **uždaviniai**, išvardinti įžangoje, buvo įgyvendinti.

Padarytos išvados bei rekomendacijos:

- Daugiausia testavime atrinktų nuotraukų (96%) pasiūlytam tiesinimo modeliui pavyko ištiesinti nustačius MPN parametro reikšmę lygią 50 ir 100. Tačiau, kai MPN lygus 100, modelis pasiūlė net 435 ištiesintus nuotraukų variantus.

- Tiesinimo modelis 92% nuotraukų ištiesino, kai MPN lygus 10, o 90% testavime dalyvavusių nuotraukų ištiesino su parametro MPN reikšme lygia 5. Kadangi tarp šių bandymų rezultatų skirtumas tik per vieną pavykusią ištiesinti nuotrauką, pasiūlymas yra naudoti tiesinimo modelį nustačius MPN reikšmę lygią 5, nes tuomet buvo sugeneruota 56-iais ištiesintais nuotraukų variantais mažiau, nei kai MPN lygu 10.

- Mažiausiai nuotraukų, 76% testavime dalyvavusių, tiesinimo modeliui pavyko ištiesinti su MPN reikšme 1, tačiau buvo pasiūlyta tik po vieną tiesinimo egzempliorių kiekvienai nuotraukai. MPN parametrą lygų 1 tinka naudoti su nuotraukomis, kuriose ryški horizonto linija, nes šiuo atveju modelis pritaiko didžiausius Gauso Filtro parametrus, todėl daug rastų linijų, yra atmesta (apie pirmas tris išvadas plačiau skyrelyje 5.2).

- Tik 26% nuotraukų ištiesino RotNet modelis „Archives“, tačiau jam apmokyti buvo naudotas $\sim$8 kartus mažesnis duomenų rinkinys nei rinkinys naudotas apmokyti RotNet „Random Pictures“ modelį ir $\sim$15 kartų mažesnis už rinkinį naudotą „All Pictures“ modeliui apmokyti (žr. skyrelį 6.2).

- 86% testavime dalyvavusių nuotraukų pavyko ištiesinti RotNet modeliui „All Pictures“. Šiam modeliui pavyko ištiesinti dvi vieninteles nuotraukas, kurių neištiesino tiesinimo modelis, kai MPN reikšmė buvo nustatyta 100 (plačiau skyrelyje 6.3).

- Jeigu yra poreikis gauti tik vieną ištiesintą nuotraukos variantą, geriau naudoti RotNet modelį „All Pictures“ nei tiesinimo modelį, nes RotNet modeliui pavyko ištiesinti 4-iomis testinėmis nuotraukomis daugiau nei tiesinimo modeliui. Tačiau jeigu siekiama ištiesinti kuo daugiau nuotraukų, verta naudoti tiesinimo modelį su MPN reikšme lygia 5.

Ateities darbų gairės

Ateities darbų sąraše gali būti tiesinimo modelio algoritmo veikimo (angl. *performance*) patobulinimas, dabar nuotraukos saugomos ir trinamos, jeigu galimas variantas netinka, galima įgyvendinti nuotraukos saugojimą, kai randamas tinkamiausias variantas (tinkami Gauso filtro parametrais).

Daugiau duomenų rinkinių skirtų RotNet tinklo apmokymui, su kurias būtų galima tiesinti nuotraukas. Šiam darbe trūko duomenų rinkinių su saulėlydžio/saulėtekių nuotraukomis, kuriose matomas ryškus horizontas, pagal kurį būtų galima tiesinti nuotraukas. Taip pat objektų, kaip langai, durys, ar pan., kuriose taip pat aptinkama tiesi linija, kuri galėtų atspindėti menamą horizontą.

Literatūros šaltiniai

[1] Ruba Anas, Hadeel A Elhadi, and Elmustafa Sayed Ali. Impact of edge detection algorithms in medical image processing. *World Scientific News*, 118:129–143, 2019.

[2] Franck Bettinger. *Probabilistic Hough transform*, 2006.

[3] MissingLink blog. *Fully Connected Layers in Convolutional Neural Networks: The Complete Guide*, 2020.

[4] Ajay Kumar Boyat and Brijendra Kumar Joshi. A review paper: noise models in digital image processing. *arXiv preprint arXiv:1505.03489*, 2015.

[5] John Canny. *A Computational Approach to Edge Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986.

[6] Patrick Brückner Daniel Saez. *RotNet*, 2019.

[7] Microsoft Azure Docs. *Install NVIDIA GPU drivers on N-series VMs running Linux*, 2019.

[8] OpenCV documentation. *Changing Colorspaces*, 2020.

[9] OpenCV documentation. *Feature Detection*, 2020.

[10] OpenCV documentation. *Hough Line Transform*, 2020.

[11] OpenCV documentation. *Image Filtering*, 2020.

[12] OpenCV documentation. *Reading and Writing Images and Video*, 2020.

[13] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[14] Stephanie Glen. *What are Robust Statistics?*, 2018.

[15] Google. *Index of / aroshan/index$_f iles/Dataset_{p}itOrlManh/zippedimages$*, 2014.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Atul Kang. *First-order Derivative kernels for Edge Detection*, 2019.

[18] Aakash Kaushik. *Understanding ResNet50 architecture*, 2020.

[19] Erwin Kreyszig. *Advanced engineering mathematics*. John Wiley & Sons, 2010.

[20] Lampros K. Michalis Lambros S. Athanasiou, Dimitrios I. Fotiadis. *Plaque Characterization Methods Using Optical Coherence Tomography*. 2017.

[21] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International journal of computer vision*, 30(2):117–156, 1998.

[22] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[23] Merriam-Webster. *Gradient*, 2020.

[24] David Peterson. *Using Horizontal Lines in Your Photography*, 2016.

[25] Prabhu. *Understanding of Convolutional Neural Network (CNN) — Deep Learning*, 2018.

[26] A. Walker E. Wolfart. R. Fisher, S. Perkins. *Feature Detectors – Sobel Edge Detector*, 2003.

[27] Adrian Rosebrock. *Zero-parameter, automatic Canny edge detection with Python and OpenCV*, 2015.

[28] Ankit Sachan. *Detailed Guide to Understand and Implement ResNets*, 2019.

[29] Daniel Saez. *Correcting Image Orientation Using Convolutional Neural Networks*, 2017.

[30] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, 2018.

[31] Sofiane Sahir. *Canny Edge Detection Step by Step in Python — Computer Vision*, 2019.

[32] Linda Shapiro. George stockman department of computer science michigan state university east lansing, mi stockman@ pixel. cps. msu. edu. pages 305–307.

[33] Anubhav Sinha. *Image Dataset*, 2020.

[34] Michael W Spratling. A neural implementation of the hough transform and the advantages of explaining away. *Image and Vision Computing*, 52:15–24, 2016.

[35] Scott E Umbaugh. *Digital image processing and analysis: human and computer vision applications with CVIPtools*. CRC press, 2010.

[36] Maria V Valueva, NN Nagornov, Pave A Lyakhov, Georgiy V Valuev, and Nikolay I Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020.

[37] Dr. Vincentas Vobolevičius. *Statistinė duomenų analizė*, 2020.

[38] Peter Welinder. *Pasadena Buildings 2010*, 2009.

Priedai

A. „Sobel“ po pirmo testo, nutriukšminus originalą

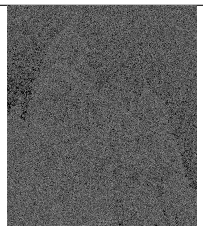
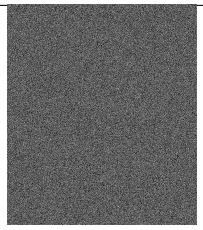
Žemiau pav. 31 pateikiamas įrodymas, kad netgi nuėmus triukšmą nuo orginalios nuotraukos (naudojant Gauso filtrą, kai $K = 7$, $\sigma = 60$), tiesių linijų, atitinkančių matomas briaunas paveikslėlyje, neaptikta.

31 pav. Pirmo testo rezultatai su „Sobel“ algoritmu, nutriukšminus. Nuotraukos iš asmeninio archyvo.

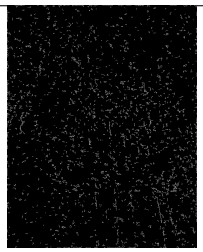
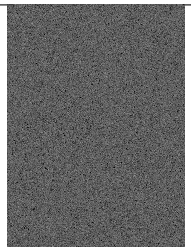
B. „Canny“ testų rezultatai su kitomis nuotraukomis

Žemiau pateikiami testavimo modelio rezultatai su kitais tyrime naudotais paveiksliukais, kai kraštų algoritmas - „Canny“, linijų radimo parametrų švelninimo žingsnis per vieną vienetą, nuotrauka Nr. 1.

10 lentelė. Antro ir trečio testų linijų rezultatai su nuotrauka Nr. 1., kraštų aptikimo algoritmas - „Canny“, parametrų švelninimas per 1vnt.

Testo nr.	Triukš. tipas	„Canny“	„Hough“	Tik. „Hough“
2	NS			
	DIP 1			
	DIP 2			
3	NS		Linijų nerado!	Linijų nerado!
	DIP 1		Linijų nerado!	Linijų nerado!
	DIP 2			

11 lentelė. Ketvirto ir penkto testų linijų rezultatai su nuotrauka Nr. 1., kraštų aptikimo algoritmas - „Canny“, parametrų švelninimas per 1vnt.

Testo nr.	Triukš. tipas	„Canny“	„Hough“	Tik. „Hough“
4	NS		Linijų nerado!	Linijų nerado!
	DIP 1		Linijų nerado!	Linijų nerado!
	DIP 2			
5	NS			Linijų nerado!
	DIP 1		Linijų nerado!	Linijų nerado!
	DIP 2			

Žemiau pateikiami testavimo modelio rezultatai su kitais tyrime naudotais paveiksliukais, kai kraštų algoritmas - „Canny“, linijų radimo parametrų švelninimo žingsnis 10%.

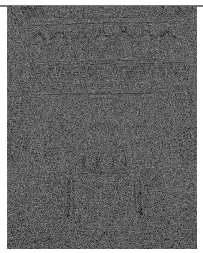
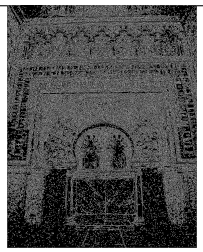
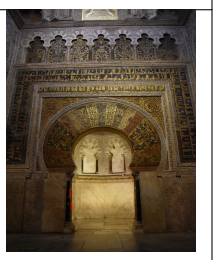
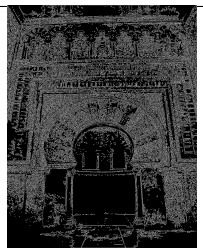
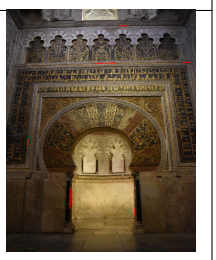
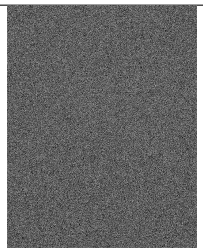
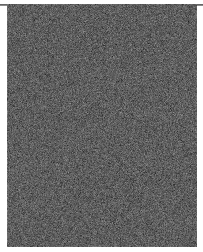
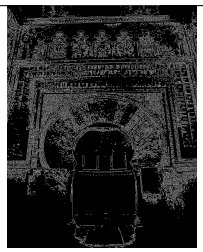
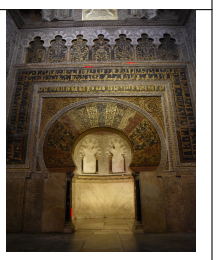
„Hough“

Linijų nerado!

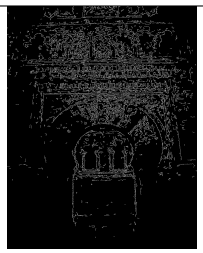
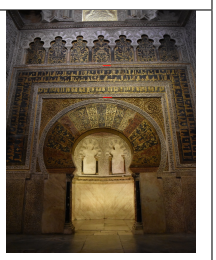
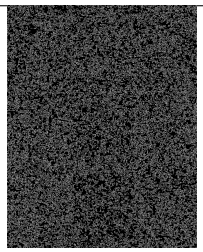
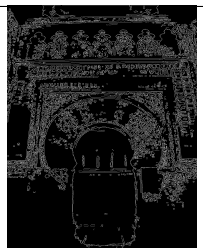
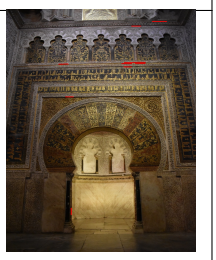
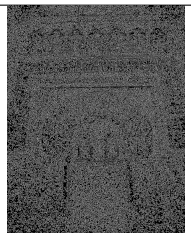
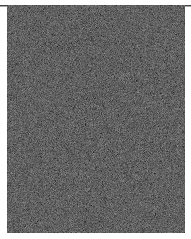
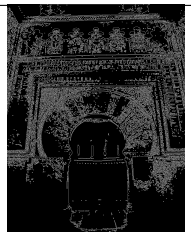
Tikimybinė „Hough“

32 pav. Rezultatai pirmo testo su patobulintais linijų uždėjimo algoritmais. Nuotrauka nr. 5. Kraštų aptikimo algoritmas - „Canny“

12 lentelė. Antro ir trečio testų linijų rezultatai su nuotrauka Nr. 5., kraštų aptikimo algoritmas - „Canny“.

Testo nr.	Triukš. tipas	„Canny“	„Hough“	Tik. „Hough“
2	NS		Linijų nerado!	Linijų nerado!
	DIP 1		Linijų nerado!	
	DIP 2		Linijų nerado!	
3	NS		Linijų nerado!	Linijų nerado!
	DIP 1		Linijų nerado!	Linijų nerado!
	DIP 2		Linijų nerado!	

13 lentelė. Ketvirto ir penkto testų linijų rezultatai su nuotrauka Nr. 5., kraštų aptikimo algoritmas - „Canny“.

Testo nr.	Triukš. tipas	„Canny“	„Hough“	Tik. „Hough“
4	NS			
	DIP 1		Linijų nerado!	Linijų nerado!
	DIP 2			
5	NS		Linijų nerado!	Linijų nerado!
	DIP 1		Linijų nerado!	Linijų nerado!
	DIP 2			

C. RotNet ir sumodeliuoto tiesinimo modelio priedai

Žemiau pateikiami tiesinimo modelio ir RotNet modelio gauti rezultatai.

14 lentelė. Tiesinimo modelio bandymo rezultatai, nuotraukų ID nuo 1 iki 40, kai MPN = 10.

Nuot. ID	TTK	Pasuktų nuot. sk.	Kampų skirtumai nuo TTK	Gauso filtro par.
1	7	1	5	Be Gauso filtro
2	-8	2	0, 1	$K = 5.\ \sigma = 0$
3	-10	3	0, 1, 2	$K = 5.\ \sigma = 30$
4	3	1	3	Be Gauso filtro
5	-5	7	0, 4, 10, 14, 17, 41, 45	$K = 5.\ \sigma = 30$
6	2	2	0, 18	$K = 7.\ \sigma = 0$
7	4	4	0, 9, 12, 13	$K = 5.\ \sigma = 0$
8	-7	5	0, 2, 19, 20, 21	$K = 5.\ \sigma = 0$
9	5	7	1, 8, 11, 12, 18, 19, 20	$K = 5.\ \sigma = 0$
10	5	3	1, 2, 3	$K = 5.\ \sigma = 30$
11	-6	3	1, 2, 4	$K = 5.\ \sigma = 30$
12	15	2	0, 1	Be Gauso filtro
13	4	1	1	$K = 5.\ \sigma = 0$
14	3	3	0, 1	$K = 5.\ \sigma = 0$
15	10	4	0, 2, 5, 6	$K = 5.\ \sigma = 0$
16	6	2	0, 1	$K = 5.\ \sigma = 0$
17	-4	2	0, 1	$K = 7.\ \sigma = 30$
18	-7	1	0	$K = 5.\ \sigma = 30$
19	4	1	2	$K = 5.\ \sigma = 30$
20	5	2	1, 2	$K = 5.\ \sigma = 30$
21	7	1	0	$K = 5.\ \sigma = 0$
22	-6	2	0, 38	$K = 5.\ \sigma = 0$
23	6	1	0	$K = 7.\ \sigma = 0$
24	4	2	30, 31	Be Gauso filtro
25	3	2	0, 1	Be Gauso filtro
26	7	1	12	$K = 7.\ \sigma = 30$
27	-6	2	0, 1	$K = 5.\ \sigma = 0$
28	-5	1	0	$K = 5.\ \sigma = 30$
29	-9	1	0	$K = 5.\ \sigma = 30$
30	5	6	2, 3, 4, 7	$K = 5.\ \sigma = 0$
31	8	2	32, 48	$K = 7.\ \sigma = 30$
32	-6	1	14	$K = 5.\ \sigma = 30$
33	-4	2	1, 3	Be Gauso filtro
34	-3	2	0, 1	$K = 7.\ \sigma = 30$
35	-6	3	1, 4, 5	$K = 5.\ \sigma = 30$
36	6	2	0, 1	$K = 5.\ \sigma = 0$
37	5	10	1, 2, 3, 4, 5, 6, 7	Be Gauso filtro
38	3	10	1, 3, 9, 10, 11, 21, 22, 25, 27, 30	$K = 7.\ \sigma = 0$
39	-3	10	0, 2, 3, 4, 7, 11, 18, 30, 32, 33	$K = 5.\ \sigma = 0$

15 lentelė. Tiesinimo modelio bandymo rezultatai, nuotraukų ID nuo 40 iki 50, kai MPN = 10.

Nuot. ID	TTK	Pasuktų nuot. sk.	Kampų skirtumai nuo TTK	Gauso filtro par.
40	-5	7	0, 1, 27, 28, 36, 38	$K = 5.\ \sigma = 0$
41	-3	3	1, 2, 47	Be Gauso filtro
42	-3	3	0, 3	$K = 5.\ \sigma = 0$
43	6	3	0, 22, 28	$K = 5.\ \sigma = 0$
44	5	10	3, 5, 8, 25, 26, 28, 30, 31, 36, 37	Be Gauso filtro
45	3	1	0	$K = 5.\ \sigma = 0$
46	5	10	0, 1, 2, 3, 4, 6, 12, 13, 17, 22	$K = 7.\ \sigma = 30$
47	7	3	0, 15, 22	Be Gauso filtro
48	5	6	4, 5, 6, 9, 10, 11	Be Gauso filtro
49	3	4	1, 2, 3, 6	$K = 5.\ \sigma = 30$
50	2	1	1	Be Gauso filtro

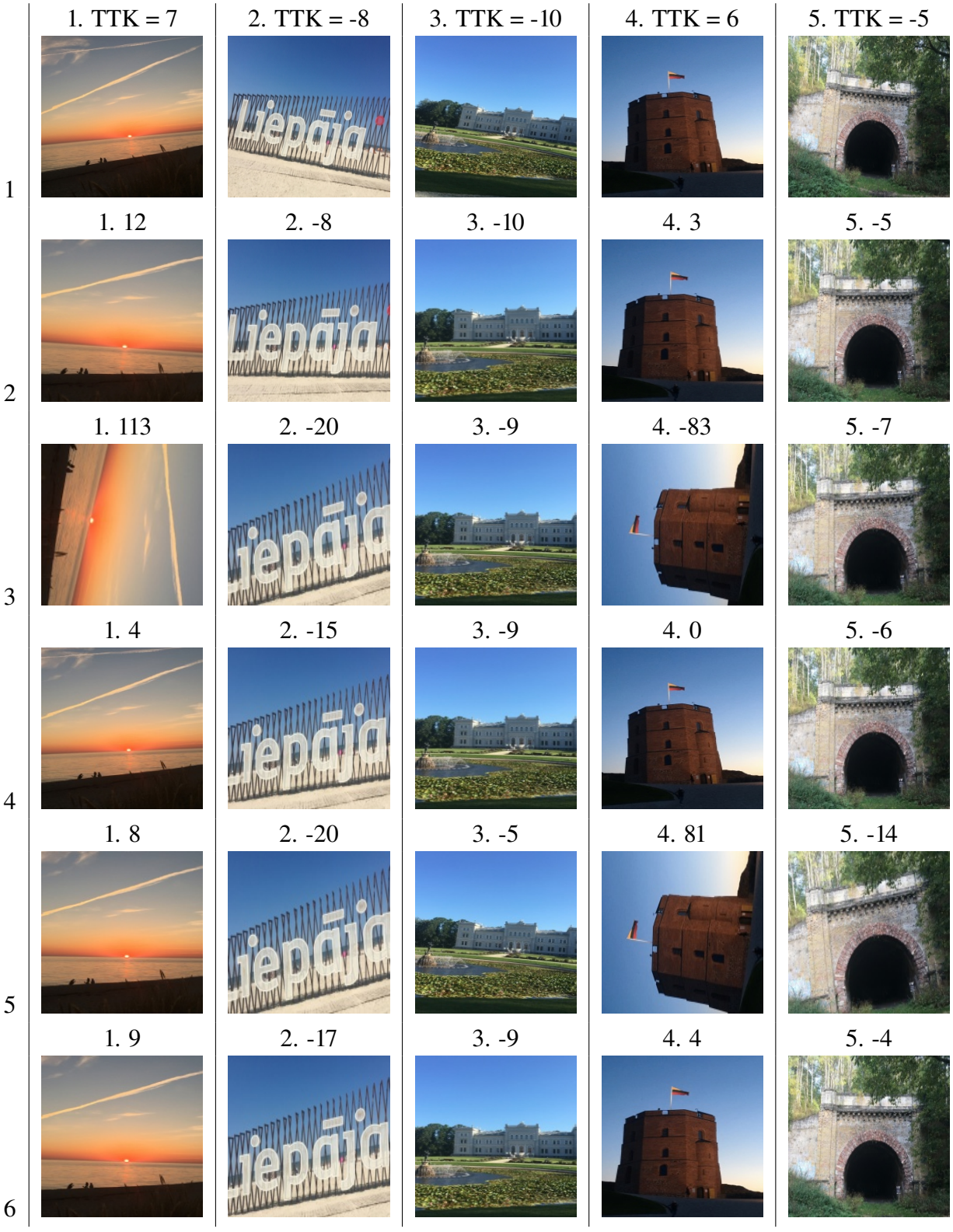
16 lentelė. RotNet modelių ir tiesinimo modelio, kai MPN lygu 10 (kampo reikšmė artimiausia TTK), pasuktų nuotraukų kampų reikšmės lyginant su TTK reikšme. Nuotraukų ID nuo 1 iki 25

ID	TTK	„StreetView"	„RandomPictures"	„Archive"	„All Pictures“	Tiesinimo
1	7	113	4	8	9	12
2	-8	-20	-15	-20	-17	-8
3	-10	-9	-9	-5	-9	-10
4	3	-83	0	-81	4	3
5	-5	-7	-6	-14	-4	-5
6	2	-3	0	6	4	2
7	4	2	-4	8	2	4
8	-7	-8	-4	-13	-9	-7
9	5	7	4	1	4	6
10	-5	-4	0	-38	-3	-4
11	-6	-6	-4	-2	-6	-7
12	15	14	12	14	14	15
13	4	40	-5	-148	85	3
14	3	-3	4	-2	1	3
15	-10	-9	-98	-2	-9	-10
16	6	175	9	146	4	6
17	-4	-3	0	8	-4	-4
18	-7	-4	176	79	172	-7
19	4	-85	2	-41	2	6
20	5	7	4	40	5	6
21	7	176	9	0	4	7
22	-6	-7	-4	0	-6	-6
23	6	98	-90	-37	-84	6
24	4	98	6	-50	9	34
25	3	4	1	1	1	3

17 lentelė. RotNet modelių ir tiesinimo modelio, kai MPN lygu 10 (kampo reikšmė artimiausia TTK), pasuktų nuotraukų kampų reikšmės lyginant su TTK reikšme. Nuotraukų ID nuo 26 iki 50

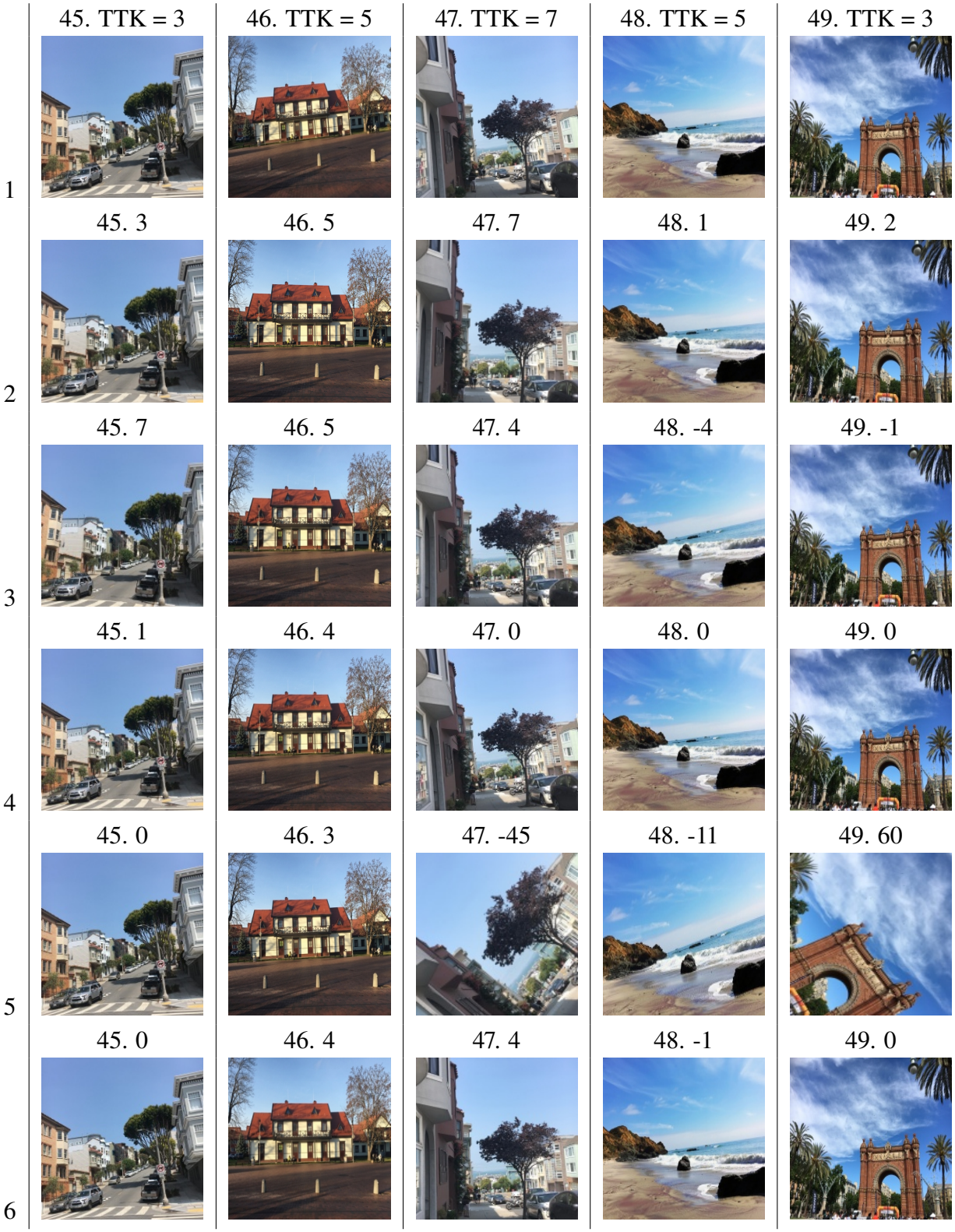
ID	TTK	„StreetView"	„RandomPictures"	„Archive"	„All Pictures“	Tiesinimo
26	7	96	4	85	9	-5
27	-6	-2	-4	-7	-4	-6
28	-5	89	-4	-83	-4	-5
29	-9	-12	-9	1	-9	-9
30	5	-11	0	-10	0	7
31	8	100	4	8	99	-24
32	-6	-9	-4	179	-1	8
33	-4	-60	-1	10	-1	-5
34	-3	-6	0	-49	-1	-3
35	-6	3	-4	-7	-6	-5
36	6	9	4	50	4	6
37	5	-81	2	-50	0	6
38	3	-89	4	41	0	4
39	-3	86	4	60	0	-3
40	-5	-9	-4	-75	-6	-5
41	-3	94	90	88	90	-4
42	-3	100	0	-37	0	-3
43	6	5	0	15	3	6
44	5	-86	-89	-31	1	2
45	3	7	1	0	0	3
46	5	5	4	3	4	5
47	7	4	0	-45	4	7
48	5	-4	0	-11	-1	1
49	3	-1	0	60	0	2
50	2	-5	0	-40	90	1

Toliau pateikiami kiti tiesinimo tyrime dalyvavusių nuotraukų ištiesinimo rezultatai naudojant RotNet modelius ir tiesinimo modelį. Žemiau parodytos ne visos testavime dalyvavusios nuotraukos, likusius rezultatus galima rasti MIF talpykloje.



33 pav. Atrinktos pirmos penkios iškreiptos nuotraukos (eilutės numeris 1, virš nuotraukos TTK reikšmė) lyginamos su tiesinimo modelio, kai MPN lygu 10, gautais rezultatais (eilutės numeris 2) bei su RotNet modelių gautomis nuotraukomis: „Street View Model" (eilutės numeris 3), „Random Pictures Model" (eilutės numeris 4), „Archives Model“ (eilutės numeris 5) ir „All Pictures Model“ (eilutės numeris 6). Virš nuotraukos nurodytas nuotraukos numeris modelio parinkto tiesinimui laipsnio reikšmė.

34 pav. Atrinktos iškreiptos nuotraukos ID nuo 10 iki 15 (eilutės numeris 1, virš nuotraukos TTK reikšmė) lyginamos su tiesinimo modelio, kai MPN lygu 10, gautais rezultatais (eilutės numeris 2) bei su RotNet modelių gautomis nuotraukomis: „Street View Model" (eilutės numeris 3), „Random Pictures Model" (eilutės numeris 4), „Archives Model“ (eilutės numeris 5) ir „All Pictures Model“ (eilutės numeris 6). Virš nuotraukos nurodytas nuotraukos numeris modelio parinkto tiesinimui laipsnio reikšmė.



35 pav. Atrinktos iškreiptos nuotraukos ID nuo 45 iki 50 (eilutės numeris 1, virš nuotraukos TTK reikšmė) lyginamos su tiesinimo modelio, kai MPN lygu 10, gautais rezultatais (eilutės numeris 2) bei su RotNet modelių gautomis nuotraukomis: „Street View Model" (eilutės numeris 3), „Random Pictures Model" (eilutės numeris 4), „Archives Model“ (eilutės numeris 5) ir „All Pictures Model“ (eilutės numeris 6). Virš nuotraukos nurodytas nuotraukos numeris modelio parinkto tiesinimui laipsnio reikšmė.