Sylinterin muotoisen kappaleen vaipan määrittäminen videosekvenssistä

Pekka Vienonen
Sovelletun matematiikan
Syventävien opintojen tutkielma
Matematiikan laitos
Joensuun Yliopisto
1997

Sisällysluettelo

1 Kolmiulotteisen kappaleen projektio kuvatasolla 4
1.1 Silmäkoordinaatisto 4
1.2 Perspektiiviprojektiot 5
2 Kuvaus vaipalta kuvatasolle ja takaisin 9
2.1 Digitaalinen kuva 9
2.2 Linssiprojektion käänteiskuvausta vastaava menetelmä 9
2.3 Skaalaustermin f määrittäminen 11
2.4 Sylinterin etäisyyden määrittäminen 12
2.5 Etäisyyden vaikutus kuvaan 13
2.6 Sylinterin korkeuden määrittäminen 13
2.7 Yksittäisen kuvan venytys (vaipan tasokuva) 14
3 Vaipan kuvan muodostaminen 19
3.1 Kuvien venytys 19
3.2 MATLAB -algoritmi kuvien venytykselle 20
3.3 Kuvien välisen siirtymän laskeminen 23
3.4 Siirtymän tarkan arvon approksimointi 24
3.5 MATLAB -algoritmi kuvien välisen siirtymän laskemiselle . . 26
3.6 Kuvien liittäminen yhdeksi kuvaksi 28
3.7 MATLAB -algoritmi kuvien yhdistämiseen 29
4 Aidon videokuvan käsittely 31
4.1 Digitaalisen kuvan suodatus (*filtering*) 31
4.2 Kohteen rajaaminen taustastaan 33
4.3 Varjostuksen poistaminen kuvasta 35
4.4 Alipikselisiirto suodatusoperaationa 36
4.5 Lopullinen vaipan tasokuva 36
4.6 Loppusanat 39

Kuvat

1.1.1 Silmäkoordinaatisto. 4
1.2.1 Kuution perspektiiviprojektio. 6
1.2.2 Kuution linssiprojektio. 7
2.2.1 Sylinterin kuvan muodostuminen kuvatasolle. 10
2.3.1 Skaalaustermin määritys 11
2.5.1 Etäisyyden vaikutus projektiokuvaan. 13
2.6.1 Sylinterin korkeuden laskeminen. 14
2.7.1 Matriisin K venytys matriisiin V. 18
3.1.1 Kuvasekvenssi loittonevasta pyörivästä sylinteristä. 19
3.1.2 Algoritmin tuottama venytettyjen kuvien sekvenssi. 20

3.4.1 Interpolaatiopinnan leikkaus arvolla y=0. 25
3.7.1 Kuvien yhdistäminen. 30
4.1.1 Keskiarvosuodatus eri kokoisilla maskeilla. 32
4.1.2 Lohkon koon vaikutus harmaasävydilaatioon. 33
4.2.1 Purkin rajaaminen häiriöisestä taustasta. 34
4.3.1 Varjostuksen poiston eri vaiheet. 35
4.3.2 Venytetyt kuvat varjostuksen poiston jälkeen. 36
4.5.1 Alkeiskuvien keskiarvo. 37
4.5.2 Alkeiskuvien mediaani yhdistettynä kuvana. 38
4.5.3 Osasuurennos mediaanikuvasta. 38

Tiivistelmä

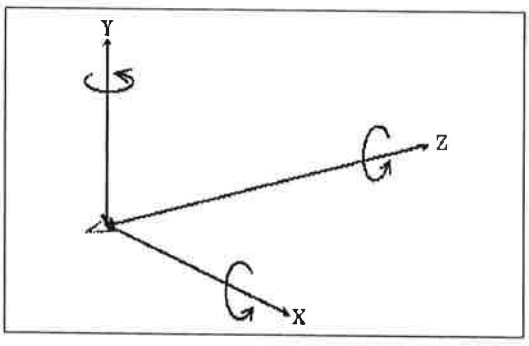
Tutkielma käsittelee automaattiseen laadunvalvontaan liittyvää sovellusta, jossa videokameran kuvien perusteella määritetään sylinterin muotoisen kappaleen vaipan kuva. Kappaleena voitaisiin ajatella olevan esimerkiksi pullo tai purkki, joka pyörii kameran edessä. Tarkoituksenani on ollut tutkia, miten ja kuinka hyvin vaipan kuva saadaan konstruoitua videosekvenssin eli kuvasarjan perusteella. Teorian pohjalta tutkielmassani vaihe vaiheelta kehitettävät algoritmit on esitetty Matlab-ympäristöön tarkoitettuna koodina. Menetelmää on mahdollista soveltaa kaikkiin sylinterinmuotoisiin kohteisiin, joista halutaan kartoittaa pinnan kuvaa tarkasti.

1. Kolmiulotteisen kappaleen projektio kuvatasolla

Jotta kolmiulotteista kappaletta voitaisiin hahmottaa kaksiulotteisesta kuvasta, on tiedettävä, kuinka kuva kolmiulotteisesta kappaleesta on syntynyt. Tarkastellaan kameralla kuvattua näkymää. Kameran toimintaperiaatteen mukaan linssi suorittaa perspektiiviprojektion näkymän pisteille. Mielivaltaisesta kaksiulotteisesta kuvasta on mahdotonta määrittää kohteen oikeita mittoja, tai eri kohteiden keskinäisiä mittasuhteita, kun etäisyydet kamerasta ovat tuntemattomia. Toisaalta, jos tunnetaan joitain kohteen mittoja, tai ominaisuuksia suhteessa muihin kohteisiin, saadaan myös etäisyysinformaatiota, sillä tunnettua on, että kaukana olevat kohteet näyttävät pienemmiltä, kuin lähellä olevat. Eli toisin sanoen, jos tiedetään, että kuva esittää kahta saman kokoista esinettä, mutta toinen näyttää pienemmältä, voidaan päätellä, että se on kauempana.

1.1. Silmäkoordinaatisto

Määritelmä 1.1.1. *Silmäkoordinaatisto (Eye Coordinate System [2]). Jos kolmiulotteisessa avaruudessa sijaitsee johonkin suuntaan katsova silmä, niin sellaista avaruuden $\mathbb{R}^3$ kantaa, jonka koordinaattiakselit on valittu siten, että x-akseli suuntautuu silmästä vaakasuoraan oikealle, y-akseli silmästä pystysuoraan ylös ja z-akseli silmästä katsesuuntaan, kutsutaan* ***silmäkoordinaatistoksi.***



Kuva 1.1.1: Silmäkoordinaatisto.

Jotta mielivaltaisesta kappaleesta saadaan esitettyä sitä vastaava projektiokuva, on kappaleen pisteet muunnettava vastaamaan silmäkoordinaatiston pisteitä. Tähän päästään **origon siirrolla** ja **akseleiden kierroilla**. Oletetaan, että katsoja sijaitsee avaruuden $\mathbb{R}^3$ pisteessä $s = (s_x, s_y, s_z)$ ja katsesuunnan määräävät kulmat ovat , $\theta_x \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ $\theta_y \in [0, 2\pi]$ ja kallistuskulma on $\theta_z \in [-\pi, \pi]$, missä θ_x vastaa kiertoa x-akselin, θ_y y-akselin ja θ_z z-akselin ympäri (ks. kuva 1.1.1). Tällöin mielivaltaista pistettä (x, y, z) vastaavat silmäkoordinaatit (x_s, y_s, z_s) ovat

$$\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = R_x R_y R_z \begin{pmatrix} x - s_x \\ y - s_y \\ z - s_z \end{pmatrix}, \tag{1.1.1}$$

missä kiertomatriisit R_x, R_y ja R_z ovat

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{pmatrix}.$$

$$R_y = \begin{pmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

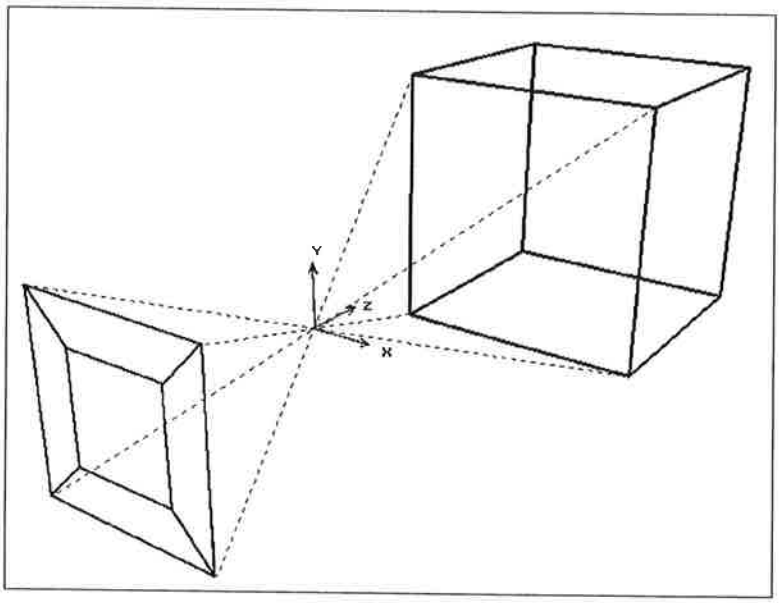
1.2. Perspektiiviprojektiot

Kolmiulotteisen näkymän perspektiiviprojektio on kuvaus $\mathbb{R}^3 \to \mathbb{R}^2$, jossa näkymään kuuluvat pisteet projisoidaan tasoon niin, että syntyneessä kuvassa on syvyysvaikutelma, eli kaukana olevat kohteet ovat pienempiä kuin lähellä olevat. Käytännössä projisointi suoritetaan esittämällä kolmiulotteisen näkymän pisteet **silmäkoordinaatistossa** ja valitsemalla kuvatasoksi xy-tason suuntainen taso.

Määritelmä 1.2.1. *(Yleinen perspektiiviprojektio, drafting perspective [3]). Silmäkoordinaatistossa esitetyn pisteen $p = (x_p, y_p, z_p)$, perspektiiviprojisoitu kuvapiste $p' = \left(x'_p, y'_p, z'_p\right)$ saadaan kaavasta*

$$\left(x'_p, y'_p, z'_p\right) = \frac{f}{z_{p,}} \left(x_p, y_p, z_{p,}\right), \tag{1.2.1}$$

missä vakio f määrää kuvatason sijainnin, eli on niin sanottu skaalaustermi. (Huomautus: kuvapisteen z-koordinaatti $z'_p = f, \forall p \in \mathbb{R}^3$, eli kaikki kuvapisteet sijaitsevat xy-tason suuntaisessa tasossa $z = f$)



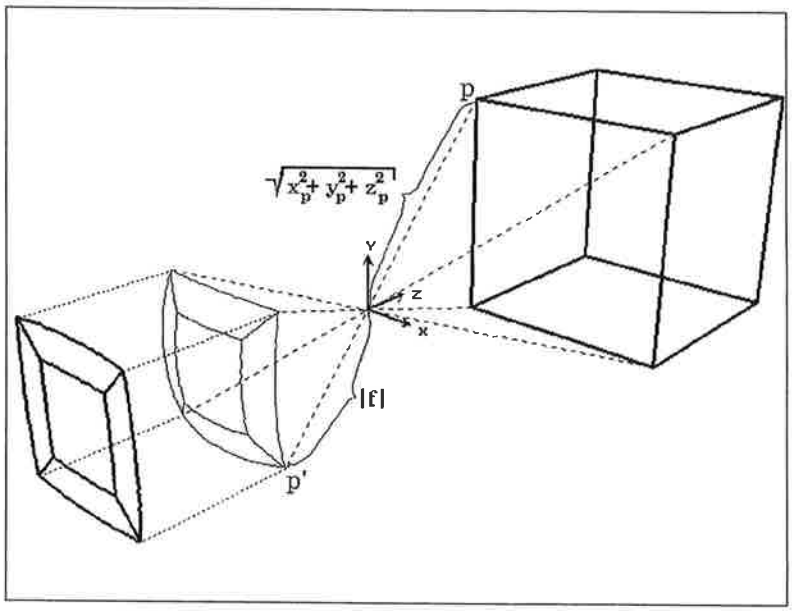
Kuva 1.2.1: Kuution perspektiiviprojektio.

Määritelmä 1.2.2. *(Linssiprojektio, true perspective [3]). Silmäkoordinaatistossa esitetyn pisteen $p = (x_p, y_p, z_p)$ pallopinnalle projisoidun kuvapisteen p' koordinaatit $\left(x'_p, y'_p, z'_p\right)$ saadaan kaavasta*

$$\left(x'_p, y'_p, z'_p\right) = \frac{f}{\sqrt{x_p^2 + y_p^2 + z_p^2}} \left(x_p, y_p, z_p\right) \tag{*}$$

Nyt skaalaustermi f vastaa pallon sädettä. (Huomautus: koordinaatit x'_p ja y'_p ilmoittavat kuvapisteen paikan kuvatasolla, ja asettamalla $z'_p = 0, \forall p$, saadaan yhdensuuntaissiirto xy-tasolle). Toisaalta, kuvapisteet saadaan suoraan xy-tasolle kirjoittamalla projisointiyhtälö (*) muotoon

$$p' = (x_p, y_p, z_p) \begin{pmatrix} \frac{f}{\sqrt{x_p^2+y_p^2+z_p^2}} & 0 & 0 \\ 0 & \frac{f}{\sqrt{x_p^2+y_p^2+z_p^2}} & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{1.2.2}$$



Kuva 1.2.2: Kuution linssiprojektio.

Vertaamalla kahta edellä esitettyä perspektiiviprojektiota keskenään, huomataan kuvien 1.2.1 ja 1.2.2 perusteella, että linssiprojektio tuottaa kuviin laajakulmalla otetuista valokuvista tutun kalansilmä -efektin, mikä tunnetaan linssivääristymänä, ja johtuu siitä, että kuvan reunoilla olevat kohteet ovat kauempana kamerasta kuin kuvan keskellä olevat (*true perspective*). Kameran muodostamaa kuvaa muistuttama kuva kolmiulotteisesta näkymästä voidaan siis luoda linssiprojektion avulla.

Yleistä perspektiiviprojektiota (*drafting perspective*) käytetään, kun halutaan tuottaa katseltavaksi tarkoitettu kuva kolmiulotteisesta näkymästä. Käytännön esimerkkinä erilaiset graafiset suunnitteluohjelmat. Vastaavasti, jos halutaan mallintaa kameran toimintaa, ja tuottaa kuvia, jotka muistuttavat kameran muodostamaa kuvaa, on käytettävä linssiprojektiota, tai vastaavaa perspektiiviprojektiota, joka kuvaa suorat viivat käyriksi lukuunottamatta kuvan keskipisteen kautta kulkevia suoria.

Mallinnettaessa kolmiulotteista maailmaa kameran tuottaman kuvan perusteella on kolmiulotteisen mallin konstruoinnissa luontevaa lähteä liikkeelle linssiprojektiosta ja etsiä sille käänteinen operaatio, joka kuvaisi kaksiulotteisen kuvan kuvapisteet kolmiulotteiseen maailmaan.

2. Kuvaus vaipalta kuvatasolle ja takaisin

2.1. Digitaalinen kuva

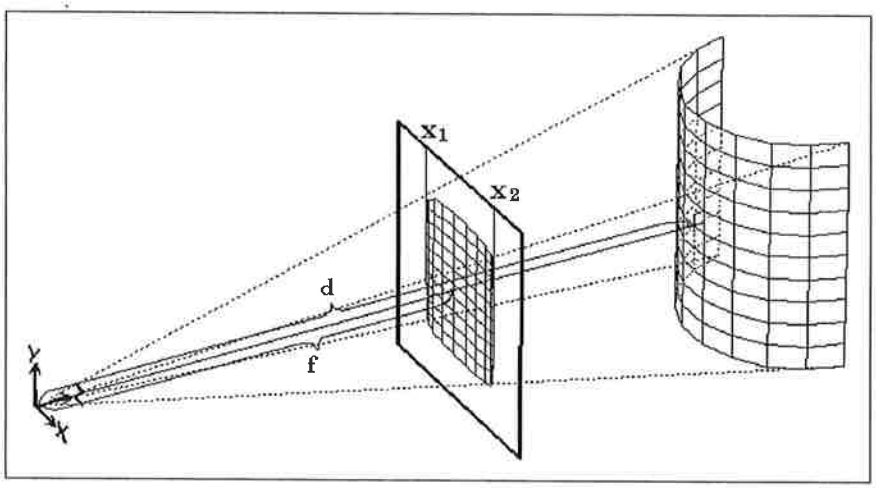
Tästä eteenpäin kuvalla tarkoitetaan digitaalista, esimerkiksi CCD-kameran tuottamaa harmaasävykuvaa. Digitaalinen kuva on muodostunut pikseleistä, eli kuva-alkioista. Jokaisella pikselillä on tietty harmaasävy, ja eri harmaasävyjen lukumäärä vaihtelee riippuen kuhunkin pikseliin käytettävien bittien määrästä, käytännössä useimmiten sävyjä on 256, mutta pienemmälläkin värien määrällä saadaan hyvä kuva, kunhan resoluutio on riittävän suuri.

Digitaalisen kuvan voidaan ajatella olevan matriisi, jossa pikselit ovat matriisin alkioita. Jos alkion lukuarvo ilmoitetaan kokonaislukuna, puhutaan indeksoidusta kuvasta, jossa alkion lukuarvo viittaa käytettävän väripaletin johonkin väriin. Kun kuvaa on tarkoitus käsitellä matemaattisilla operaatioilla on käytännöllisempää skaalata alkioiden arvot välille $[0, 1]$, vieläpä niin, että nolla tarkoittaa mustaa ja ykkönen valkoista väriä. Näin esitettyä kuvamatriisia sanotaan intensiteettikuvaksi. Jatkossa käsiteltävät kuvat ovat intensiteettikuvia.

2.2. Linssiprojektion käänteiskuvausta vastaava menetelmä

Tarkastellaan kuvaa, joka esittää sylinteriä suoraan sivulta. Jokainen pikseli kuvassa vastaa tiettyä aluetta sylinterin vaipalla. Kuvan keskialueilla olevat pikselit vastaavat pienempää aluetta kuin pikselit kuvan reunoilla. Jos tarkoituksena on tuottaa tasokuva sylinterin vaipasta edellä mainitun kuvan perusteella, ei se onnistu pelkästään johtamalla linssiprojektiolle käänteiskuvaus, jolla kuvataan sylinteriä esittävän kuvan pikselit niitä vastaaviin paikkoihin vaippaa esittävään kuvaan. Osa vaipan pisteistä jää näin meneteltynä käsittelemättä, johtuen juuri reunalla ja keskellä olevista eri kokoisista alueista. Jos taas tiivistämme vaipan kuvaa niin ettei aukkoja synny, hävitämme samalla keskialueilta informaatiota, kun pikselit ikään kuin peittävät toinen toisiaan.

Kuinka sylinteriä esittävästä kuvasta saadaan konstruoitua vaipasta näkyvää osaa esittävä tasokuva hävittämättä yhtään pikseli-informaatiota? Lyhyesti menetelmä on seuraavanlainen: Määritetään kuvan perusteella sylinterin sijainti silmäkoordinaatistossa, eli paikannetaan kolmiulotteisesti kuvattu kohde. Tämän jälkeen projisoidaan saadussa paikassa sijaitsevan sylinterin pinnan pisteitä kuvatasolle, ja poimitaan saaduista paikoista kuvatasolta pinnan pisteitä vastaavat harmaasävyt vaippaa kuvaavaan matriisiin. Toisin sanoen simuloimme kuvan muodostumista kuvitteellisella kameralla kuvitteellisesta sylinteristä tilanteessa, joka pyrkii vastaamaan todellisuutta.

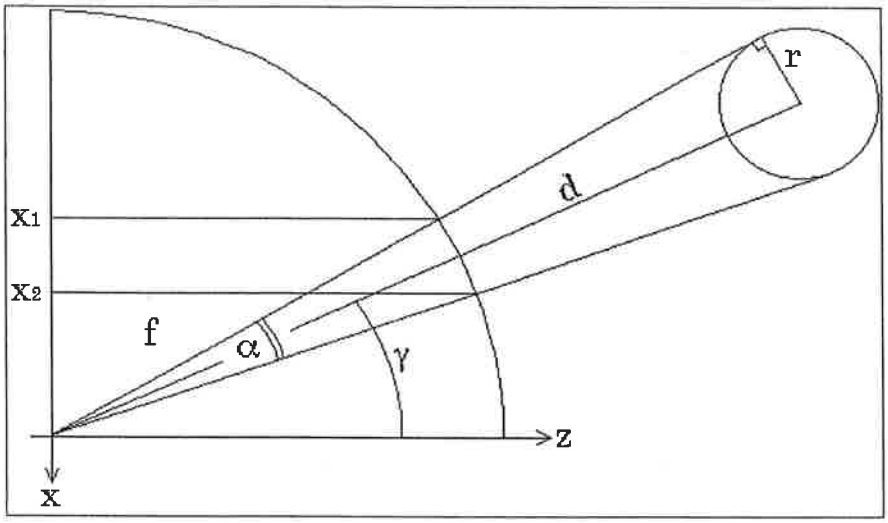


Kuva 2.2.1: Sylinterin kuvan muodostuminen kuvatasolle.

Oletetaan sylinterin säde r tunnetuksi ja oletetaan lisäksi, että sylinteri on pystysuorassa ja sylinterin keskipiste sijaitsee samalla korkeudella kuin kamera. Nyt, jos skaalaustermi f tunnetaan, voidaan projektiokuvan vasemman ja oikean reunan sijainnin perusteella määrittää sylinterin etäisyys d kamerasta. Vastaavasti, jos tunnetaan etäisyys d, voidaan määrittää skaalaustermi f. Kun kyseessä on videosekvenssi, voi etäisyys kamerasta vaihdella eri kuvien välillä, mutta f on vakio (olettaen, että kuvien välillä ei ole käytetty zoomia). Näin ollen on tiedettävä kohteen etäisyys vähintään yhdessä videokuvassa, jotta se voidaan määrittää muihin kuviin.

2.3. Skaalaustermin f määrittäminen

Oletetaan, että sylinterin säde on r ja etäisyys kamerasta on d. Olkoon lisäksi projektiokuvan vasen reuna suoralla x_1 ja oikea reuna suoralla x_2 (kuten kuvassa 2.2.1). Merkitään lisäksi muuttujalla γ sylinterin pystyakselin ja



Kuva 2.3.1: Skaalaustermin määritys

näköakselin z välistä kulmaa. Kuvassa 2.3.1 on vastaava tilanne esitetty ylhäältä katsottuna. Sylinteri näkyy kameraan kulmassa

$$\alpha = 2 \arcsin\left(\frac{r}{d}\right) \tag{2.3.1}$$

Skaalaustermille f saadaan nyt kaksi eri esitystä 2.3.2, josta edelleen saadaan ratkaistua kulma γ

$$\begin{cases} f = -\frac{x_1}{\sin\left(\gamma+\frac{\alpha}{2}\right)} \\ f = -\frac{x_2}{\sin\left(\gamma-\frac{\alpha}{2}\right)} \end{cases} \tag{2.3.2}$$

$$\Rightarrow \quad \frac{x_1}{\sin\left(\gamma+\frac{\alpha}{2}\right)} = \frac{x_2}{\sin\left(\gamma-\frac{\alpha}{2}\right)}$$

$$\Rightarrow \quad \frac{\sin\left(\gamma+\frac{\alpha}{2}\right)}{\sin\left(\gamma-\frac{\alpha}{2}\right)} = \frac{x_1}{x_2}$$

$$\Rightarrow \quad \frac{\sin\gamma\cos\frac{\alpha}{2}+\cos\gamma\sin\frac{\alpha}{2}}{\sin\gamma\cos\frac{\alpha}{2}-\cos\gamma\sin\frac{\alpha}{2}} = \frac{x_1}{x_2}$$

$$\Rightarrow \quad \gamma = \arctan\left(\frac{(x_1+x_2)\sin\alpha}{(\cos\alpha+1)(x_1-x_2)}\right),$$

mikä on siis sylinterin poikkeama näköakselista. Skaalaustermille f saadaan arvo sijoittamalla γ jompaan kumpaan yhtälöistä 2.3.2:

$$f = -\frac{x_1}{\sin\left(\arctan\left(\frac{(x_1+x_2)\sin\alpha}{(\cos\alpha+1)(x_1-x_2)}\right)+\frac{\alpha}{2}\right)} \tag{2.3.3}$$

2.4. Sylinterin etäisyyden määrittäminen

Olkoon sylinterin säde r ja projisoinnissa käytetty skaalaustermi f. Olkoon lisäksi projektiokuvan vasen reuna suoralla x_1 ja oikea reuna suoralla x_2. Kuvasta 2.3.1 nähdään, että kulmalle α pätee

$$\alpha = \arcsin\frac{x_2}{f} - \arcsin\frac{x_1}{f}$$

Sijoittamalla näin saatu arvo kaavaan 2.3.1 saadaan sylinterin etäisyydelle d johdettua kaava

$$2\arcsin\left(\frac{r}{d}\right) = \arcsin\frac{x_2}{f} - \arcsin\frac{x_1}{f}$$

$$\Rightarrow \quad \frac{r}{d} = \sin\left(\frac{\arcsin\frac{x_2}{f}-\arcsin\frac{x_1}{f}}{2}\right)$$

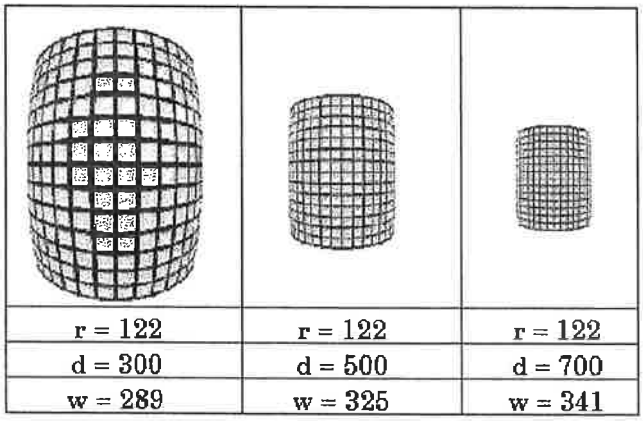
$$\Rightarrow \quad d = \frac{r}{\sin\left(\frac{\arcsin\frac{x_2}{f}-\arcsin\frac{x_1}{f}}{2}\right)} \tag{2.4.1}$$

Näin saadulla kaavalla voidaan siis määrittää r -säteisen sylinterin etäisyys kussakin videosekvenssin kuvassa, kun tunnetaan skaalaustermi f ja kuvan vasen ja oikea reuna kuvatasolla.

2.5. Etäisyyden vaikutus kuvaan

Sylinterin etäisyys kamerasta vaikuttaa oleellisesti kuvassa olevaan vaippaa koskevaan informaatioon. Mitä lähempänä kameraa sylinteri on, sitä pienempi osa vaipasta näkyy. Etäisyyden tunteminen on välttämätöntä, jotta saadaan lasketuksi sylinteristä näkyvän vaipan osan leveys ja korkeus. Vaipasta näkyvän osan leveys muuttuu etäisyyden funktiona.

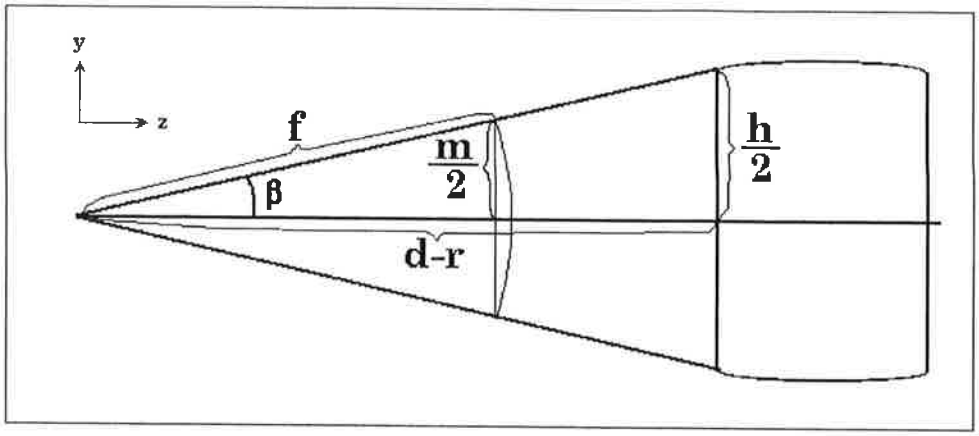
Lisäksi, jos kohde on "riittävän lähellä" kameraa, sylinteristä muodostuneen kuvan muoto muuttuu etäisyydestä riippuen enemmän tai vähemmän "pallomaiseksi". Etäisyyden kasvaessa vaipasta näkyvä osuus lähestyy puolivaippaa ja projisoituneen kuvan muoto suorakulmiota. Kuvassa 2.5.1 on havainnollistettu etäisyyden d vaikutusta vaipasta näkyvän osan leveyteen w. Kuvasta havaitaan myös sylinterin kuvan muodon muuttuminen etäisyyden vaikutuksesta.



Kuva 2.5.1: Etäisyyden vaikutus projektiokuvaan.

2.6. Sylinterin korkeuden määrittäminen

Sylinterin projektiokuvan korkeuden perusteella on mahdollista laskea sylinterin korkeus h. Olkoon kuvan korkeus m. Lisäksi tunnetaan skaalaustermi f ja etäisyys d, sekä sylinterin säde r.Kuva 2.6.1 esittää vastaavaa tilannetta



Kuva 2.6.1: Sylinterin korkeuden laskeminen.

sivulta katsottuna. Johdetaan kaava korkeudelle h lähtien liikkeelle kulman β tangentista:

$$\frac{\sin\beta}{\cos\beta} = \frac{\frac{h}{2}}{d-r}$$

$$\Rightarrow \quad \frac{mf}{2f\sqrt{f^2-\left(\frac{m}{2}\right)^2}} = \frac{h}{2\,(d-r)}$$

$$\Rightarrow \quad h = \frac{m\,(d-r)}{\sqrt{f^2-\left(\frac{m}{2}\right)^2}}. \tag{2.6.1}$$

2.7. Yksittäisen kuvan venytys (vaipan tasokuva)

Kun on selvitetty kuvasta r-säteisen sylinterin etäisyys d, korkeus h, sekä kulma γ, eli poikkeama näköakselista, voidaan johtaa kuvan esittämälle vaipan osalle, pinnalle $V \subset \mathbb{R}^3$ parametriesitys silmäkoordinaatistossa:

$$\begin{cases} x = d\cos\left(\frac{\pi}{2}+\gamma\right) + r\cos\left(\gamma+\frac{3\pi}{2}+\theta\right) \\ y = t \\ z = d\sin\left(\frac{\pi}{2}+\gamma\right) + r\sin\left(\gamma+\frac{3\pi}{2}+\theta\right) \end{cases}, \begin{cases} \theta \in [-\cos^{-1}\frac{r}{d}, \arccos\frac{r}{d}] \\ t \in [-\frac{h}{2}, \frac{h}{2}] \end{cases}$$

Sijoituksella $s = r\theta$, saadaan parametriesitys sievennettyä muotoon:

$$\begin{cases} x = -d\sin\gamma - r\sin\left(\gamma + \pi + \frac{s}{r}\right) \\ y = t \\ z = d\cos\gamma + r\cos\left(\gamma + \pi + \frac{s}{r}\right) \end{cases}, \begin{cases} s \in [-r\arccos\frac{r}{d}, r\arccos\frac{r}{d}] \\ t \in [-\frac{h}{2}, \frac{h}{2}] \end{cases} \tag{2.7.1}$$

Tämä kuvaa vaipasta näkyvää osaa vastaavan alueen st-tasolta silmäkoordinaatistoon.

On huomattava, että puhuttaessa kuvan leveyksistä ja korkeuksista tarkoitetaan pikseleiden lukumäärää. Kuitenkin luonnolliset etäisyydet, kuten sylinterin etäisyys kamerasta ja sylinterin säde voivat olla vaikkapa millimetrejä tai metrejä, ja seurauksena edellisestä myös vaipan näkyvää osaa vastaava alue st-tasolla on mitoiltaan luonnollisessa koossa. Tarkoitus ei ole kuitenkaan muodostaa kohteesta luonnollisen kokoista pikselikuvaa - kohteenahan voi olla vaikkapa vesitornin kylki, tai mikroskooppikuva sylinterin muotoisesta bakteerista.

Oleellista tuotettavan kuvan koon määräytymisessä on käytettävän videokuvan sisältämän informaation määrä, eli kuvapisteiden lukumäärä. Seuraavaksi määritellään vaipan tasokuvaa esittävälle matriisille mielekäs dimensio ja suoritetaan muuttujien s ja t arvoalueille dimension määräämät tasaväliset jaot, toisin sanoen määrätään tuotettavan kuvan resoluutio.

Tuotettavan kuvan koko pikseleinä määräytyy käsiteltävästä projektiokuvasta K siten, että jos projektiokuva K on $m \times n$ -matriisi, niin valitaan tuotettavaksi tasokuvaksi $m^* \times n^*$ -matriisi V, missä

$$\begin{cases} m^* = \left[m\frac{\sqrt{(d-r)^2 + \left(\frac{h}{2}\right)^2}}{(d-r)} \right] \\ n^* = \left[\frac{2r\arccos\frac{r}{d}}{h} \cdot m\frac{\sqrt{(d-r)^2 + \left(\frac{h}{2}\right)^2}}{(d-r)} \right] \end{cases} \tag{2.7.2}$$

Perusteluna edellä esitetylle seuraavanlainen intuitiivinen tarkastelu: Sylin-

terin kuvan avaaminen tasoon on eräänlainen venytys, jossa alkuperäisen kuvan reuna-alueita venytetään. Pystysuuntaista venymistä vastaava kerroin saadaan tarkastelemalla perspektiivistä johtunutta kutistumista. Venytyskerroin pystysuunnassa on sylinterin yläosan etäisyys kamerasta suhteessa keskiosan etäisyyteen kamerasta (vertaa kuva 2.6.1). Vaakasuuntainen venytyskerroin saadaan kertomalla pystysuuntainen kerroin vaipan tasokuvan leveyden ja korkeuden suhteella.

Nyt sijoittamalla $i = 1 + \left(\frac{t}{h} + \frac{1}{2}\right)(m^* - 1)$ ja $j = 1 + \left(\frac{s}{2r \arccos \frac{r}{d}} + \frac{1}{2}\right)(n^* - 1)$ saadaan kaava 2.7.1 muotoon:

$$\begin{cases} x = -d \sin\gamma - r \sin\left(\gamma + \pi + 2\left(\frac{j-1}{n^*-1} - \frac{1}{2}\right) \arccos\frac{r}{d}\right) \\ y = \left(\frac{i-1}{m^*-1} - \frac{1}{2}\right) h \\ z = d \cos\gamma + r \cos\left(\gamma + \pi + 2\left(\frac{j-1}{n^*-1} - \frac{1}{2}\right) \arccos\frac{r}{d}\right) \end{cases}, \quad \begin{cases} i = 1, ..., m^* \\ j = 1, ..., n^* \end{cases}.$$

Näin saatu kaava antaa sylinterin vaippaa kuvaavan $m^* \times n^*$ -matriisin V jokaiselle alkiolle v_{ij} sitä vastaavan paikan (x, y, z) silmäkoordinaatistossa. Projisoimalla näin saadut silmäkoordinaatiston pisteet p_{ij} linssiprojektiolla xy-tasoon saadaan vaipan pisteitä vastaavat kuvatason koordinaatit (x', y'). Kuvatason pisteitä (x', y') vastaavat matriisin K rivi- ja sarakeindeksit (i', j') saadaan merkitsemällä $i' = \left[\frac{m}{2} + y'\right]$ ja $j' = \left[\frac{n}{2} + x'\right]$. Kun sijoitetaan $v_{ij} = k_{i'j'}$ saadaan jokaiselle vaipan pisteelle v_{ij} poimittua arvo alkuperäisestä kuvasta K.

Esimerkki 2.7.1. *Olkoon* (254×171) *-matriisi* K *sivukuva sylinteristä, jonka säde* $r = 12.2$, *ja olkoon kuvassa sylinterin ja kameran välinen etäisyys* $d = 40$. *Muodostetaan matriisi* V, *joka kuvaa sylinteristä näkyvän vaipan osaa tasossa. Aluksi lasketaan skaalaustermi* f ,

$$\begin{aligned} f &= -\frac{x_1}{\sin\left(\arctan\left(\frac{(x_1+x_2)\sin\alpha}{(\cos\alpha+1)(x_1-x_2)}\right)+\frac{\alpha}{2}\right)} \\ &= -\frac{-85}{\sin\left(\arctan(0)+\arcsin\left(\frac{12.2}{40}\right)\right)} \\ &= \frac{85\cdot 40}{12.2} \approx 279. \end{aligned}$$

ja sylinterin korkeus h ,

$$h = \frac{m(d-r)}{\sqrt{f^2-\left(\frac{m}{2}\right)^2}} \Rightarrow h = \frac{254\cdot(40-12.2)}{\sqrt{279^2-127^2}} \approx 28.4.$$

Seuraavaksi lasketaan muodostettavan kuvan V *dimensio* $(m^* \times n^*)$,

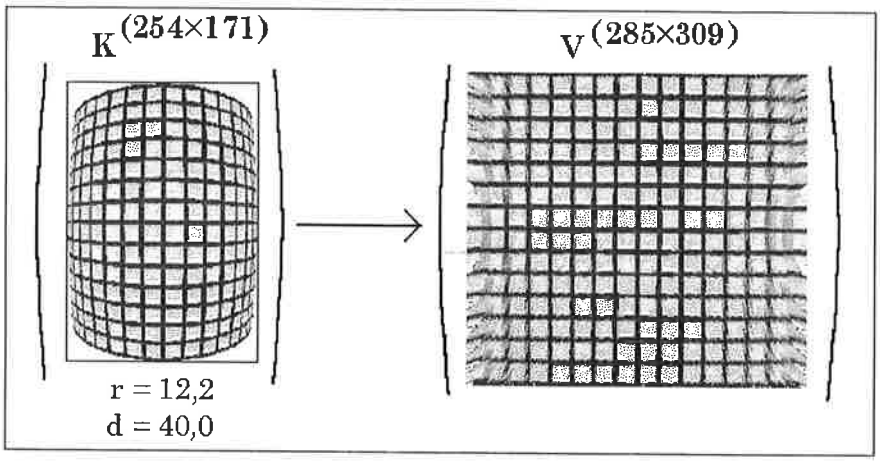
$$\begin{cases} m^* = \left[254\cdot\frac{\sqrt{27.8^2+14.2^2}}{27.8}\right] = 285 \\ n^* = \left[\frac{24.4\cdot\arccos 0.305}{28.4}\cdot 254\cdot\frac{\sqrt{27.8^2+14.2^2}}{27.8}\right] = 309 \end{cases}$$

ja nyt kuva $V^{(285\times 309)}$ *saadaan muodostetuksi asettamalla* $V(i,j) = K(i',j')$, *kun*

$$\begin{cases} i' = \left[\frac{254}{2}+\frac{280}{\sqrt{x_p^2+y_p^2+z_p^2}}y_p\right] \\ j' = \left[\frac{171}{2}+\frac{280}{\sqrt{x_p^2+y_p^2+z_p^2}}x_p\right] \end{cases}, \text{ missä}$$

$$\begin{cases} x_p = 12.2\cos\left(\frac{3\pi}{2}+2\left(\frac{j-1}{308}-\frac{1}{2}\right)\arccos 0.305\right) \\ y_p = \left(\frac{i-1}{284}-\frac{1}{2}\right)\cdot 28.3 \\ z_p = 40+12.2\sin\left(\frac{3\pi}{2}+2\left(\frac{j-1}{308}-\frac{1}{2}\right)\arccos 0.305\right) \end{cases},$$

kun $i = 1,\ldots,285$ *ja* $j = 1,\ldots,309$.



Kuva 2.7.1: Matriisin K venytys matriisiin V.

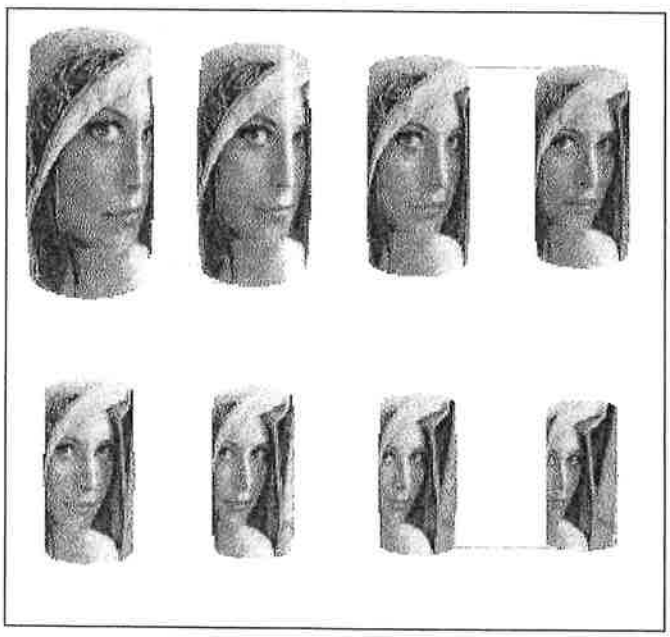
Kuvasta 2.7.1 nähdään, kuinka venytys tuottaa mittasuhteiltaan oikean tasokuvan sylinterin vaipasta. Kuvan laatu pysyy melko hyvänä lähes reunoille asti. Reuna-alueille ei tarkempaa kuvaa voi muodostaa yhden kuvan perusteella, kun muistetaan, että alkuperäisen kuvan pikselin arvo on sille alueelle projisoituneiden pisteiden keskiarvo. Kuvan laatua voi parantaa käyttämällä venytyksessä suurempaa kuvakokoa, jolloin pyöristysten aiheuttama suhteellinen virhe on pienempi, kun lopuksi kutistetaan saatu kuva haluttuun kokoon.

3. Vaipan kuvan muodostaminen

Jos videosekvenssi esittää kameran edessä pyörivää sylinteriä, on mahdollista rakentaa sekvenssin kuvia yhdistämällä sylinterin vaippaa esittävä kuva. Aluksi on jokaiselle sekvenssin kuvalle suoritettava edellä esitetyn kaltainen venytys, jolloin sekvenssin kuvat yhdenmuotoistuvat. Sen jälkeen tasokuvat yhdistetään yhdeksi kuvaksi sovittamalla ne päällekkäin ja laskemalla kuvien keskiarvokuva.

3.1. Kuvien venytys

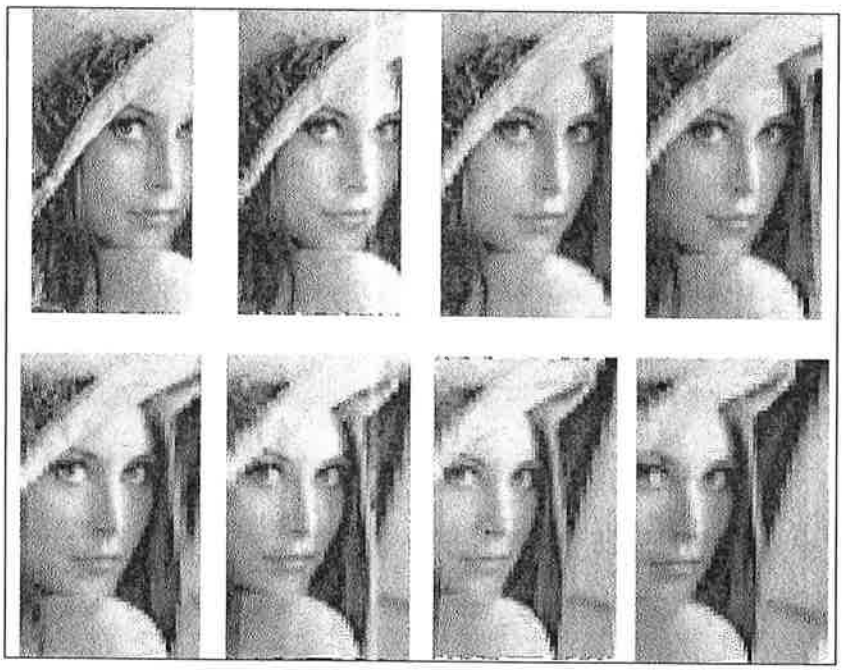
Menetelmä perustuu edellisessä luvussa esitettyyyn yksittäisen kuvan venytykseen pienin muutoksin. Koska tarkoituksena on myöhemmin yhdistää venytetyt kuvat päällekkäin, tulee niiden olla keskenään samassa mittakaavassa. Venytyksessä ei siis lasketa jokaiselle kuvalle omaa kuvakokoa $m^* \times n^*$, vaan käytetään kaikissa venytyksissä samaa kuvakorkeutta, joka valitaan vastaamaan suurinta yksittäistä kuvaa. Kuvakorkeuden määrää siis se video-



Kuva 3.1.1: Kuvasekvenssi loittonevasta pyörivästä sylinteristä.

sekvenssin kuva, jossa sylinteri on lähinnä kameraa. Tällöin kuvat, joissa kohde on kauempana suurenevat suhteessa enemmän. Kaikki tuotettavat kuvat ovat keskenään saman korkuisia, mutta eri levyisiä johtuen etäisyyden vaikutuksesta näkyvän osan suuruuteen.

Kuvassa 3.1.1 on esitetty simuloitu kuvasarja pyörivästä sylinteristä, joka loittonee kamerasta. Kun tällaisesta sekvenssistä tuotetaan vastaava tasokuvien sekvenssi saadaan kuvasarja, joka on esitetty kuvassa 3.1.2.



Kuva 3.1.2: Algoritmin tuottama venytettyjen kuvien sekvenssi.

3.2. MATLAB -algoritmi kuvien venytykselle

Seuraavassa esitetään MATLAB-kielinen koodi, joka suorittaa videosekvenssille edellä esitetyn kuvien venytyksen. Kuva 3.1.2 on tuotettu tässä esitettävällä ohjelmalla, kun sille on annettu syötteenä kuvan 3.1.1 esittämä sekvenssi. Pyöristysvirheiden vähentämiseksi algoritmi käyttää venytyksessä nelinkertaista kuvakokoa, ja venytyksen jälkeen syntynyt kuva kutistetaan

neljänteen osaan korvaamalla aina neljä pikseliä niiden keskiarvolla. Usein kuvankäsittelyoperaatiot kannattaa suorittaa alkuperäistä kuvaa suuremmalle kuvalle, minkä jälkeen saatu kuva kutistaan lopulliseen kokoon.

Toinen mahdollinen tapa on käyttää rivi- ja sarakeindekseinä reaalilukuja. Tällöin suoritetaan pikselille desimaaliosien suuruinen alipikselisiirto negatiiviseen suuntaan, jolloin saadaan painokertoimet kolmelle lähimmälle pikselille. Kuvamatriisiin lisätään näiden kolmen pikselin painotettu keskiarvo. Toisin sanoen oletetaan reaaliarvoisessa paikassa sijaitsevan pikselin olevan kolmen kokonaislukuindeksisen pikselin interpolaatio. Alipikselisiirtoon sekä algoritmin varjostuksen poistamista käsittelevään osaan palataan luvussa 4, joka käsittelee digitaalisen kuvan käsittelyoperaatioita.

```
function [ISOKUVA] = projisoi(SEK,lkm,d,r,sw);
% funktio 'projisoi(SEK,lkm,d,r,sw)'
% Vers. 3.0 (C)1996 P.Vienonen
%
% - oikaisee sylinterin kuvat tasoon, eli tuottaa kuvasekvenssin,
%   jossa kussakin kuvassa vaipasta nakyva osa tasossa.
%
% SEK: kuvasekvenssi, jossa framet allekkain
% lkm: sekvenssissa olevien kuvien lukumaara
%   d: kohteen etaisyys kamerasta ensimmaisessa kuvassa
%   r: purkin sade
%  sw: 0 tai 1, varjostuksen poisto off/on

% ---------- alustetaan vektoreita, apumuuttujia ja -matriiseja;
D               = zeros(1,lkm);M=zeros(1,lkm);N=zeros(1,lkm);
kor             = size(SEK,1)/lkm;
[x1,x2,y1,y2]   = leikkaa(SEK([1:kor],:));
m               = abs(y1-y2);
alfa            = pi-2*acos(r/d);
gamma           = atan((x1+x2)*sin(alfa)/((cos(alfa)+1)*(x1-x2)));
f               = (-x1/sin(gamma+alfa/2));
h               = m*(d-r)/sqrt(f^2-m^2/4);
for i=1:lkm,
  [x1,x2,y1,y2]= leikkaa(SEK(((i-1)*kor)+[1:kor],:));
  M(i)          = abs(y1-y2);
  alfa          = asin(x2/f)-asin(x1/f);
  G(i)          = atan((x1+x2)*sin(alfa)/((cos(alfa)+1)*(x1-x2)));
  D(i)          = r/cos((pi-alfa)/2);
  end
```

```
kokoy           = max(2*round(M.*sqrt((D-r).^2+(h/2)^2)./(D-r)));
kokox           = 2*round(0.5*(r*pi/h)*kokoy);
ISOKUVA         = zeros(lkm*kokoy/2,kokox/2+1);
KUVA            = zeros(kor,size(SEK,2));
alkurivi        = (0:(lkm-1))*(kokoy/2);
xc              = ones(kokoy,1)*(r*cos(pi+pi*(([1:kokox]-0.5)/kokox)));
yc              = (h*([1:kokoy]/kokoy-0.5))'*ones(1,kokox);
zc              = ones(kokoy,1)*(r*sin(pi+pi*(([1:kokox]-0.5)/kokox)));
% ---------- aloitetaan kuvien kasittely;
for l = 1:lkm,
  KUVA = SEK(((l-1)*kor)+[1:kor],:);
  origx=round(size(KUVA,2)/2);
  origy=round(size(KUVA,1)/2);
  disp(['muodostetaan kuvaa ' int2str(l) '...'])
  dx=-D(l)*sin(G(l));
  dz= D(l)*cos(G(l));
  x=xc+dx;
  y=yc;
  z=zc+dz;
  dist=sqrt(x.^2+y.^2+z.^2);
  px=origx+round(x./dist*f);
  py=origy+round(y./dist*f);
% ---------- poimitaan projektiopisteista alkiot vaipan tasokuvaan;
  alkuj=round(kokox/4*(1-0.85*acos(r/D(l))/(pi/2)));
  loppuj=kokox/2-alkuj;

  for j=(2*alkuj):2:(2*loppuj),
    for i=2:2:size(px,1),
      ISOKUVA(alkurivi(l)+i/2,j/2) =...
      (KUVA(py(i-1,j-1),px(i-1,j-1)) + KUVA(py(i,j),px(i,j))+...
      KUVA(py(i-1,j),px(i-1,j))+KUVA(py(i,j-1),px(i,j-1)))/4;
      end;
    end;
% ---------- Varjostuksen poisto (ks. Luku 4);
  if sw==1
    K=ISOKUVA(alkurivi(l)+[1:size(px,1)/2],alkuj:loppuj);
    Kf=colfilt(K,[5 5],'sliding','max');   % Gray-scale dilation
    Kf=filter2(ones(5)/25,Kf);        % Smoothing
    Kf(:,1:2)=Kf(:,3:4);Kf(:,size(Kf,2)-[0:1])=Kf(:,size(Kf,2)-[2:3]);
    K=max(Kf(:))*(K./(Kf+(Kf==0)));        % varjot pois
    K=K.*((K>=0)&(K<=1))+(K>1);            % skaalaus valille [0..1]
    ISOKUVA(alkurivi(l)+[1:size(K,1)],[alkuj:loppuj])=K;
    end;
  end;
end.
```

3.3. Kuvien välisen siirtymän laskeminen

Kun jokaiselle kuvalle on suoritettu venytys, saadaan videosekvenssi, jossa on vaipan tasokuvia. Tarkastelemalla kuva-alkioiden liikeratoja alkuperäisissä ja venytetyissä kuvissa, havaitaan, että venytettyjen kuvien välillä liikeradat ovat vaakasuoria, kun taas alkuperäisten kuvien välillä liikeradat olivat kaarevia. Lisäksi liike on tasaista koko kuva-alueella.

Kun siirtymien suuruudet kuvien välillä saadaan määritettyä, on mahdollista asettaa kuvat päällekkäin, jolloin saadaan sylinterin vaippaa esittävä kuva, johon on käytetty kaikkien kuvien informaatio. Siirtymän laskemiseksi kahden kuvan välillä poimitaan toisesta kuvamatriisista vertailupala ja etsitään sitä vastaava alue toisesta kuvamatriisista. On löydettävä jokin kriteeri, millä määrätään, kuinka hyvin kaksi kuvaa, vastaavat toisiaan.

Määritelmä 3.3.1. *Kuvien sovitus (block matching). Olkoot $m \times n$ -matriisit A ja B kaksi kuvamatriisia. Määritellään matriisien A ja B eroavuus $e(A, B)$ kaavalla:*

$$e(A,B) = \sum_{i=1}^{m}\sum_{j=1}^{n}\left(\frac{a_{ij}}{\overline{a}} - \frac{b_{ij}}{\overline{b}}\right)^2 \tag{3.3.1}$$

Toisin sanoen eroavuus e on alkioidensa keskiarvoilla normeerattujen kuvamatriisien A ja B erotusmatriisin E alkioiden neliösumma. Näin määriteltynä eroavuus e saa sitä pienempiä arvoja, mitä paremmin kuvamatriisit A ja B vastaavat toisiaan. Jakamalla A ja B alkioidensa keskiarvoilla eliminoidaan matriisien mahdollinen tummuusero. Neliösumman asemasta eroavuuden laskemisessa voidaan käyttää myös keskiarvoa, mediaania, keskihajontaa, tai vastaavaa. Sopivin kaava kustannusfunktiolle e riippuu kuvien luonteesta. Matlab-kielinen funktio eroavuuden laskemiseksi on seuraava:

```
function e=bmatch(A,B);
A=A/mean(A(:));
B=B/mean(B(:));
E = (A-B).^2;
e = sum(E(:));
end;
```

Tarkastellaan tilannetta, jossa K_1 ja K_2 ovat $m \times n$ -matriiseja, jotka esittävät pyörivän sylinterin vaipan tasokuvia eri hetkellä. Valitaan vertailupalaksi K_1^* $m^* \times n^*$ -**blokki** matriisista K_1 siten, että $K_1^* = K_1\left(i_0{:}\,(i_0 + m^*)\,, j_0{:}(j_0 + n^*)\right)$. Seuraavaksi etsitään kuvamatriisista K_2 vertailupalaa K_1^* vastaava alue etsimällä minimi luvuista

$$e_{ij} = e\left(K_1^*, K_2\left(i{:}(i + m^*), j{:}(j + n^*)\right)\right), \text{ kun } \begin{cases} i = 1, ..., m - m^* \\ j = 1, ..., n - n^* \end{cases}.$$

Olkoon tuo minimi $e_{i'j'}$. Tällöin siirtymä $(\Delta i, \Delta j)$kuvien K_1 ja K_2 välillä on

$$(\Delta i, \Delta j) = (i_0 - i', j_0 - j'). \tag{3.3.2}$$

3.4. Siirtymän tarkan arvon approksimointi

Edellä esitetyllä menetelmällä saadaan lasketuksi kuinka monta saraketta ja riviä kahden kuvan välillä on siirtymää. Todellisuudessa siirtymän suuruus on harvoin kokonaisluku, ja siirtymää onkin mahdollista approksimoida tarkemmin kuin pikselin tarkkuudella.

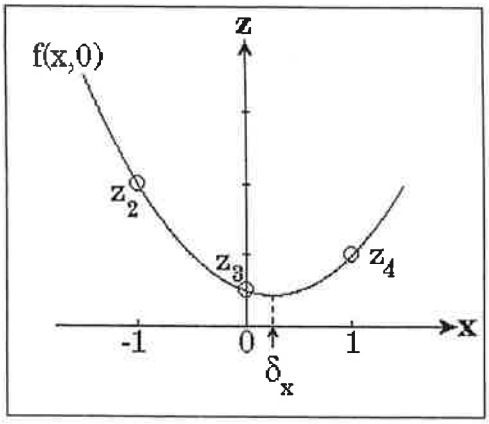
Olkoon matriisi E eroavuuslukujen e_{ij} muodostama $(m - m^*) \times (n - n^*)$ -matriisi., ja valitaan i' ja j' siten, että $e_{i'j'} = \min(e_{ij})$. Kustannusfunktiolle e on luonteenomaista, että minimin ympäristössä se käyttäytyy kuin paraboloidi. Ajatellaan matriisin E olevan otos erään kolmiulotteisen pinnan pisteitä. Approksimoidaan pintaa E sellaisella toisen asteen funktiolla $f : \mathbb{R}^2 \longmapsto \mathbb{R}$, jonka arvot origon ympäristössä vastaavat pinnan E pisteitä minimipisteessä (i', j')ja sen naapuripisteissä. Valitaan funktion f arvojoukoksi pistejoukko Z, siten, että

$$\begin{aligned} Z &= \{z_1, z_2, z_3, z_4, z_5\} \\ &= \{E(i' - 1, j'), E(i', j' - 1), E(i', j'), E(i', j' + 1), E(i' + 1, j')\}. \end{aligned}$$

Joukko Z siis sisältää näin ollen minimipisteen, sekä sen neljä naapuripistettä.

Toisen asteen pinta f, joka kulkee pisteiden $(0,-1,z_1)$, $(-1,0,z_2)$, $(0,0,z_3)$, $(0,1,z_4)$ ja $(0,1,z_5)$ kautta on

$$f(x,y) = z_{3+}\frac{(z_4-z_2)\,x+(z_2-2z_3+z_4)\,x^2+(z_5-z_1)\,y+(z_1-2z_3+z_5)\,y^2}{2}$$



Kuva 3.4.1: Interpolaatiopinnan leikkaus arvolla y=0.

Pinnan f minimipisteessä on voimassa $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y} = 0$. Joten minimipisteelle (x_0, y_0) saadaan kaava:

$$\begin{cases} \frac{\partial f}{\partial x}(x_0,y_0) = 0 \\ \frac{\partial f}{\partial y}(x_0,y_0) = 0 \end{cases}$$

$$\Rightarrow \begin{cases} \frac{(z_4-z_2)}{2} + (z_2-2z_3+z_4)\,x_0 = 0 \\ \frac{(z_5-z_1)}{2} + (z_1-2z_3+z_5)\,y_0 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} x_0 = \frac{(z_2-z_4)}{2(z_2-2z_3+z_4)} \\ y_0 = \frac{(z_1-z_5)}{2(z_1-2z_3+z_5)} \end{cases}$$

Näin ollen siis alipikselitarkka approksimaatio siirtymälle $(\delta i, \delta j)$ kaavan 3.3.2 perusteella on

$$(\delta i, \delta j) \;=\; \left(\Delta i + \frac{(z_1-z_5)}{2\,(z_1-2z_3+z_5)}, \Delta j + \frac{(z_2-z_4)}{2\,(z_2-2z_3+z_4)}\right),$$

missä $(z_1, z_2, z_3, z_4, z_5)$

$$= (E(i'-1,j'), E(i',j'-1), E(i',j'), E(i',j'+1), E(i'+1,j')).$$

3.5. MATLAB -algoritmi kuvien välisen siirtymän laskemiselle

Seuraavassa esitetään MATLAB-koodi, joka laskee venytettyjen kuvien sekvenssille siirtymät $(m \times 2)$ -matriisiin, jossa kukin rivi ilmoittaa X ja Y -siirtymän kullekin kuvalle. Pyörimissuunnasta riippuu onko siirtymä kumulatiivinen suhteessa ensimmäiseen vai viimeiseen kuvaan. Siirtymien arvot ovat aina positiivisia ja suuntana on vasemmalta oikealle, johtuen myöhemmin esitettävästä algoritmista, joka liittää sekvenssin kuvat päällekkäin huomioiden kullekin kuvalle kuvakohtaisen siirtymän.

```
function [xypaikat]=laskexy(SEK,lkm);
%function [xypaikat]=laskexy(SEK,lkm);
%Vers. 3.1 (C)1996 P.Vienonen
%Palauttaa arvonaan lkm*2 -matriisin, jossa on kullekin
%kuvalle x- ja y- siirtyma suhteessa ensimmaiseen tai
%viimeiseen kuvaan riippuen liikkeen suunnasta.
%Syotteena on sekvenssi SEK, ja kuvien lukumaara lkm.

n     = size(SEK,2);
h     = size(SEK,1)/lkm;
OrigX = round(n/2);OrigY = round(h/2);
DY = fix(h/3);DX = fix(n/6);
H=([1 2 1]'*[1 2 1])./16;
IM1 = round(4*filter2(H,SEK(h+(1:h),:)))/4;
IM2 = round(4*filter2(H,SEK(2*h+(1:h),:)))/4;
IMc = IM2(OrigY+(-DY:DY),OrigX+(-DX:DX));
%----- etsitaan ensin oletusarvo siirtymalle...
minimi=1000;
   for j = (-OrigX+DX+1):(OrigX-DX-1),
     ero = bmatch(IM1(OrigY+(-DY:DY),j+OrigX+(-DX:DX)),IMc);
     if ero<=minimi
            minimi= ero;
            jj    = j;
            end,
   end
ind=abs(jj);  % tuleva oletusarvo siirtymalle
suunta=sign(jj);if suunta==0,suunta=1;end;
```

```
DY    = fix(h/4);
DX    = fix(n/7);
yla   = OrigY-DY;
ala   = OrigY+DY;
vas   = OrigX-DX;
oik   = OrigX+DX;
summax=0;summay=0;
X=zeros(1,lkm);
Y=X;
% ----- lasketaan siirtymat...
for nr = 1:lkm-1,
   if (suunta==1) nro = nr-1;
             else nro = lkm-nr;end
   disp(['tutkitaan kuvaa ' int2str(nro+suunta)])
   IM1 = SEK(nro*h+(1:h),:);
   IM2= SEK((nro+suunta)*h+(1:h),:);
   IMc  = IM2(yla:ala,vas:oik);
   j=ind;i=0;EC=1;Emin=0;
   Emin=EC-1;
   while Emin~=EC,
         ii=i;
         jj=j;
         EC=bmatch(IM1(i+(yla:ala),j+(vas:oik)),IMc);
         EY=bmatch(IM1(i-1+(yla:ala),j+(vas:oik)),IMc);
         EA=bmatch(IM1(i+1+(yla:ala),j+(vas:oik)),IMc);
         EV=bmatch(IM1(i+(yla:ala),j-1+(vas:oik)),IMc);
         E0=bmatch(IM1(i+(yla:ala),j+1+(vas:oik)),IMc);
         Emin=min([EC EY EA EV E0]);
         if Emin==EY
               i=i-1;
            elseif Emin==EA
               i=i+1;
            elseif Emin==EV
               j=j-1;
            elseif Emin==E0
               j=j+1;
            end;
     end;
   Y(nro+1+suunta) = summay+ii+(EA-EY)/(4*EC-2*EY-2*EA);
   X(nro+1+suunta) = summax+jj+(E0-EV)/(4*EC-2*EV-2*E0);
   summax = X(nro+1+suunta);
   summay = Y(nro+1+suunta);
   end
xypaikat=[X',Y'];
end.
```

Kuvalle 3.1.2 edellä esitetty algoritmi antaa tulokseksi matriisin:

$$\begin{pmatrix} 0 & 0 \\ 7.0246 & 0.1481 \\ 13.7556 & 0.3489 \\ 21.4747 & 0.3442 \\ 27.8940 & 0.4664 \\ 35.4445 & 0.4172 \\ 42.1009 & 0.0761 \\ 48.4327 & 1.3304 \end{pmatrix}$$

3.6. Kuvien liittäminen yhdeksi kuvaksi

Kun on laskettu venytettyjen kuvien sekvenssille siirtymät, liitetään kuvat yhteen laittamalla ne päällekkäin siirtymät huomioiden ja laskemalla päällekkäin olevien kuvien keskiarvo lopulliseen kuvaan. Koska lopullisen kuvan jokainen pikseli muodostuu monen eri alkeiskuvan pikseleistä, on mahdollista suurentaa tuotettavan kuvan resoluutiota suurentamalla kutakin alkeiskuvaa vakiokertoimella, kun samalla kertoimella kerrotaan siirtymät.

Alkeiskuville on suoritettava vielä niin sanottu alipikselisiirto ennen yhdistämistä. Jos suurennuskerroin on k ja siirtymä on $\delta = (\delta i, \delta j)$ niin alipikselisiirron suuruus on $k\delta - \lfloor k\delta \rfloor$, eli alipikselisiirto vastaa siirtymän desimaaliosaa ja kuuluu näin ollen välille $[0, 1)$. Alle pikselin suuruinen siirto alkeiskuvalle saadaan aikaan suodattamalla kuva erityisellä siirtomatriisilla, joka interpoloi kuvapisteet ympäröivistä pisteistä, tästä tarkemmin luvussa 4.

Lopulliseksi kuvaksi lasketaan alkeiskuvien keskiarvo. Keskiarvon laskennassa voidaan käyttää myös painotusta siten, että alkeiskuvien reuna-alueiden pikseleitä painotetaan vähemmän, jolloin venytyksestä johtuva mahdollinen virhe pienenee.

3.7. MATLAB -algoritmi kuvien yhdistämiseen

Kuvat yhdistävä algoritmi käyttää venytettyjen kuvien sekvenssiä, sekä X ja Y -siirtymiä. Lopullisen kuvan koko määrätään kertoimella, joka ilmoittaa vaaka- ja pystysuunnassa tapahtuvan laajennuksen, eli esimerkiksi arvolla $k = 3$ kukin alkeiskuvan pikseli vastaa 3×3 pikselin suuruista aluetta lopullisessa kuvassa.

```
function [I] = kokoa(SEK,k,xypaikat);
% funktio 'kokoa(SEK,k,xypaikat)' luo vaipan
% kuvan venytetyista kuvista kertoimella k.
% SEK on venytettyjen kuvien sekvenssi,
% xypaikat on 'laskexy(SEK,lkm)'
m      = size(SEK,1)/size(xypaikat,1);
n      = size(SEK,2);
lkm    = size(xypaikat,1);
d      = round(max(xypaikat(:,2))-min(xypaikat(:,2))*k)+1;
leveys = round(max(xypaikat(:,1))*k+k*n);
I      = zeros((2*d+m)*k,leveys);
K      = I;
for nro=1:lkm,
  kuva  = SEK((nro-1)*m+(1:m),:);
  [i,j] = find(kuva(fix(m/2),:));
  kuva  = kuva(:,j);   % ei oteta mustia reunoja mukaan
  n     = size(kuva,2);
  v     = 1+floor((0:(n*k-1))/k);
  u     = 1+floor((0:(m*k-1))/k);
  kuva  = kuva(u,v);
  realX = xypaikat(nro,1)+j(1)-1;
  realY = xypaikat(nro,2);
  isoX  = floor(k*realX);isoDX=(k*realX)-isoX;
  isoY  = d*k+floor(k*realY);isoDY=(k*realY)-floor(k*realY);
  kuva  = filter2(siirto(isoDX,isoDY),kuva);
  row   = [2:(m*k-1)];col=[2:(n*k-1)];
  I(isoY+row,isoX+col)=I(isoY+row,isoX+col)+kuva(row,col);
  K(isoY+row,isoX+col)=K(isoY+row,isoX+col)+1;
  end
I = I./((K==0)+K);
end;
```

Kuva 3.7.1: Kuvien yhdistäminen.

Kuvassa 3.7.1 on esitetty sekvenssin (kuva 3.1.1) kuvien perusteella laskettu vaipan tasokuva. Vertailun vuoksi kuvassa on vasemmalla yksi kuva venytettyjen kuvien sekvenssistä (kuva 3.1.2). Kertoimena yhdistettäessä on käytetty arvoa $k = 2$, eli yhdistetyn kuvan resoluutio on kaksinkertainen venytettyyn kuvaan verrattuna.

4. Aidon videokuvan käsittely

Tähän mennessä edellä kehitettyä menetelmää on testattu ohjelmallisesti tuotetuilla sekvensseillä. Aidon videokuvan käsittelyyn liittyen on koejärjestelyissä ja lopullisessa käyttösovellutuksessa huomioitava muutamia seikkoja, ennen kuin kehitettyjä menetelmiä voidaan soveltaa videosekvenssille. Ensinnäkin sylinterin pyörimisliike kuvien välillä ei saa olla liian suurta, eikä pyörimisnopeus saa vaihdella liikaa kuvien välillä. Algoritmi on räätälöitävä tältä osin tilannetta vastaavaksi kulloisenkin käyttösovelluksen mukaan.

Videokuvassa on myös aina häiriötä, joka on pyrittävä poistamaan erilaisilla digitaalisilla suodattimilla. Kaikki tummuuserot kuvassa eivät tarkoita, että kohteessa olisi aina tekstuuria, vaan pieniä harmaasävyn vaihteluita esiintyy aina, vaikka kuvattava pinta olisi täysin tasavärinen. Häiriön lisäksi tummuuseroja aiheuttaa varjostus. Sylinterin muotoisen kappaleen pinnalle syntyy aina varjostus niin, että reuna-alueet näyttävät tummemilta kuin keskiosa. Varjostus häiritsee liikkeen määrittämistä, sillä se on ristiriidassa tekstuurin liikkeen kanssa - vaikka sylinteri pyörii, varjot eivät liiku. Varjostus on poistettava kuvasta ohjelmallisesti, sillä valaistuksella sen poistaminen olisi melko mahdotonta.

4.1. Digitaalisen kuvan suodatus (*filtering*)

Digitaalisen kuvan suodatuksella tarkoitetaan kuvankäsittelyoperaatiota, jossa suodatetun kuvan pikseleiden arvot määräytyvät alkuperäisen kuvan pikseleistä ja niiden ympäristöistä. Suodatuksen tarkoituksena on saada kuvasta poistetuksi turha informaatio, tai korostaa tarvittavaa informaatiota. Usein tarpeellinen informaatio on ikään kuin "kätkeytynyt" kuvaan, ja erilaisilla suodatusoperaatioilla se saadaan esiin.

Lineaariset suodattimet (*Linear filters*) ovat operaattoreita H, jotka voidaan esittää $(2m+1) \times (2n+1)$ -matriiseina. Suodatettu kuva P_f on alku-

peräisen kuvan P ja suodattimen, eli maskin, H konvoluutio

$$P_f = P * H.$$

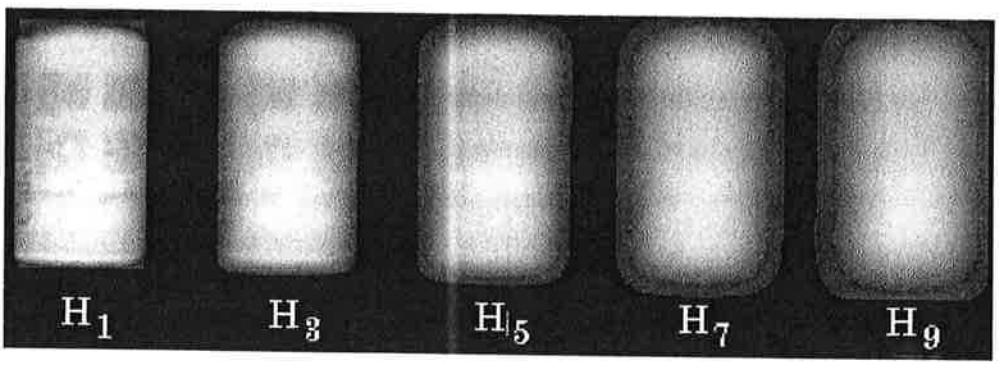
Diskreetissä tapauksessa kuvan $P \in \mathbb{R}^{m \times n}$suodatetun kuvan P_f pikselit määräytyvät maskin H alueella kulloinkin olevien pikseleiden painotetusta summasta

$$P_f(r,s) = \sum_{i=0}^{2m} \sum_{j=0}^{2n} H(1+i, 1+j) P'(r+i, s+j), \text{ kun } \begin{cases} r = 1, ..., M \\ s = 1, ..., N \end{cases} \quad (4.1.1)$$

Suodatettavan kuvan P tilalla käytetään suurempaa kuvaa P', joka on saatu lisäämällä kuvan P ylä- ja alareunaan m riviä, ja reunoille n riviä nollaalkioita. Esimerkkinä alipäästösuodatus (*low pass filtering [1]*), joka yksinkertaisimmillaan saadaan aikaan keskiarvosuodattimella (*averaging filter*)

$$H_n = \begin{pmatrix} \frac{1}{(2n+1)^2} & \cdots & \frac{1}{(2n+1)^2} \\ \vdots & \ddots & \vdots \\ \frac{1}{(2n+1)^2} & \cdots & \frac{1}{(2n+1)^2} \end{pmatrix} \in \mathbb{R}^{(2n+1)\times(2n+1)}$$

Suodatus maskilla H_n tai jollain muulla alipäästösuodattimella poistaa kuvasta korkeataajuiset komponentit (kuva 4.1.1). Alipäästösuodatusta käytetään myös kohinan vaimentamiseen kuvasta (*smoothing*).

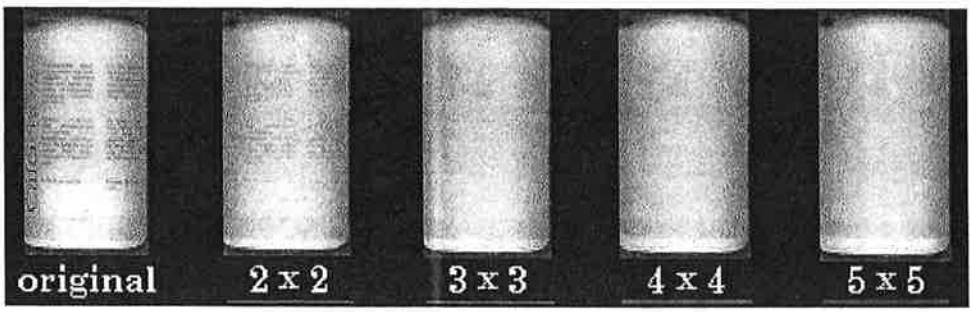


Kuva 4.1.1: Keskiarvosuodatus eri kokoisilla maskeilla.

Epälineaariset suodattimet (*rank filters, non-linear filters*) ovat operaatioita, joita ei voida esittää kerroinmatriisina H, koska ne vaativat tietyllä alueella oleville pikseleille suoritettavaa järjestelyä tai vastaavaa. Esimerkkinä harmaasävydilaatio, jossa kuvan $P \in \mathbb{R}^{M \times N}$ suodatetun kuvan P_f pikselit muodostuvat rajoitetun alueen $[2m+1 \times 2n+1]$ pikseleiden maksimiarvoista

$$P_f(r,s) = \max\{P'(r+i, s+j) \mid i \in [0,...,2m] \wedge j \in [0,...,2n]\}, \quad (4.1.2)$$

kun $r = 1,...,M$ ja $s = 1,...,N$. Kaavassa esiintyvä P' on muodostettu kuvasta P samoin, kuin edellä kaavassa 4.1.1.



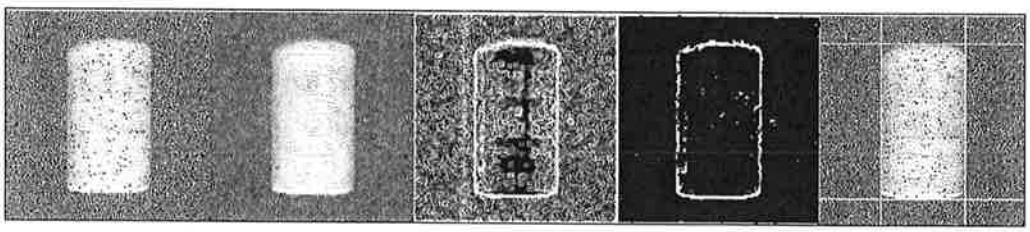
Kuva 4.1.2: Lohkon koon vaikutus harmaasävydilaatioon.

4.2. Kohteen rajaaminen taustastaan

Ohjelmallisesti luoduissa testisekvensseissä oli tasaisen musta tausta, joten kohde oli helppo rajata alimatriisiin etsimällä vain pienin ja suurin rivi- ja sarakeindeksi, jolla sijaitsee nollasta eroavia alkioita. Käytännössä tausta ei koskaan ole pelkkiä nollia, johtuen kohinasta, jota aidossa videodatassa aina on. Kynnystämällä (*thresholding*) kuva, eli asettamalla nolliksi kaikki tietyn kynnysarvon alittavat pikselit, saadaan taustan häiriöt useimmiten pois. Vaarana on, että tällöin kynnystäminen poistaa myös purkin reunoja, mikä aiheuttaa etäisyyden määrittämiseen virhettä.

Ennen kynnystystä kannattaakin kuvasta etsiä särmät. Suodattamalla kuva niin sanotuilla Sobel-operaattoreilla, G_x ja G_y (4.2.1) saadaan kuvasta esiin pysty- ja vaakasuuuntaiset särmät. Operaattorit laskevat pikselille derivaatan suhteessa naapuripisteisiin, eli G_x ja G_y ovat suunnattuja derivaattoja, ja pikselin gradientti $\nabla p = |G_x| + |G_y|$.

$$G_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}, G_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{4.2.1}$$



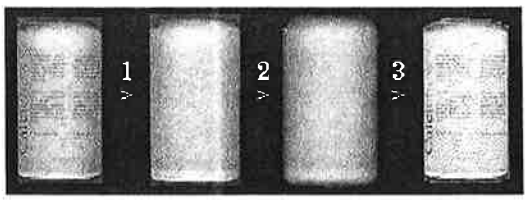
Kuva 4.2.1: Purkin rajaaminen häiriöisestä taustasta.

- Käsiteltävänä kuvana on videokuva, johon on lisätty kahta eri kohinaa, Gaussin kohinaa, sekä 'salt & pepper' -kohinaa.
- Seuraavana on kuva mediaani suodatuksen jälkeen. Mediaanisuodatus vähentää oleellisesti erityisesti 'salt & pepper' -kohinaa.
- Keskimmäinen kuva on edellisen gradientti, eli $|G_x| + |G_y|$
- Gradienttikuvalle on suoritettu kynnystys arvolla 1, eli kaikki ykköstä pienemmät arvot on nollattu.
- Viimeinen kuva on alkuperäinen kuva, johon on piirretty kynnystetystä kuvasta löydetyt rajat.

4.3. Varjostuksen poistaminen kuvasta

Tarkastellaan kuvan 4.1.2 'original' -kuvaa. Jotta saataisiin selville, mikä purkin pinnalla on varjoa, on jollain tavoin erotettava kuvasta varjostus ja muu tekstuuri. Varjostukselle on ominaista, että se muuttuu hitaasti tummasta vaaleaan ja takaisin tummaan. Näin ollen tehokkaasti alipäästösuodattamalla kuvasta jää jäljelle varjostuksen osuus. Suuri alipäästösuodatin kuitenkin levittää kuvaa taustaansa.

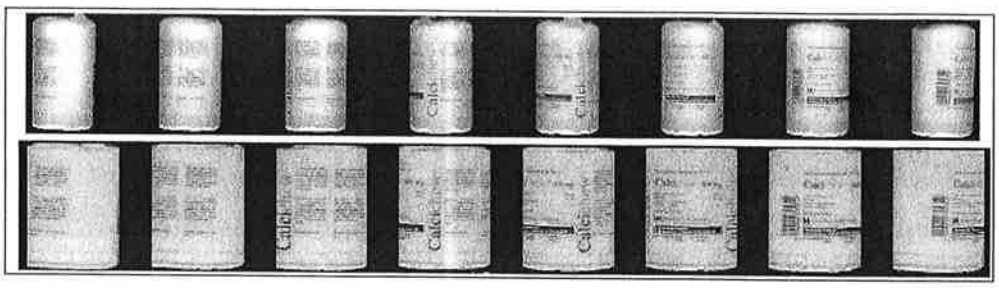
Kun kuvalle I suoritetaan ensin harmaasävydilaatio, ja sen jälkeen syntynyt kuva alipäästösuodatetaan, saadaan samea harmaa alue, jossa on jäljellä varjostuksen aiheuttama tummuusvaihtelu. Alkuperäisestä kuvasta saadaan näin löydetty varjostus poistettua jakamalla se alkioittain varjostuksen kuvalla, minkä jälkeen tulosmatriisin alkiot on vielä skaalattava uudelleen välille $[0, 1]$.



Kuva 4.3.1: Varjostuksen poiston eri vaiheet.

Kuvassa 4.3.1 on esitetty vaiheittain varjostuksen poistaminen. Vasemmalla on alkuperäinen kuva purkista.

- Vaiheessa (1) on kuvalle suoritettu harmaasävydilaatio 5×5 -kokoisella lohkolla.
- Seuraavana dilaation tuottama kuva on alipäästösuodatettu keskiarvosuodattimella H_2 (2).
- Lopuksi vaiheessa (3) on alkuperäinen kuva jaettu alkioittain alipäästösuodatetulla kuvalla ja tulosmatriisin alkiot on skaalattu välille $[0, 1]$.



Kuva 4.3.2: Venytetyt kuvat varjostuksen poiston jälkeen.

4.4. Alipikselisiirto suodatusoperaationa

Koska kuvien väliset siirtymät ovat reaalilukuja, ja rivi- ja sarakeindeksit kokonaislukuja, on alkeiskuville suoritettava siirtymän desimaaliosan suuruinen alipikselisiirto. Määritellään siirtomatriisi, jolla suodattamalla kuva siirtyy alle pikselin suuruisen matkan δ_r pikseliä alas ja δ_s pikseliä oikealle, seuraavasti

$$H(\delta_r, \delta_s) = \begin{cases} \begin{pmatrix} \delta_r & 0 & 0 \\ \delta_s - \delta_r & 1 - \delta_s & 0 \\ 0 & 0 & 0 \end{pmatrix}, \text{ kun } 0 \leq \delta_r < \delta_s \leq 1. \\ \begin{pmatrix} \delta_s & \delta_r - \delta_s & 0 \\ 0 & 1 - \delta_r & 0 \\ 0 & 0 & 0 \end{pmatrix}, \text{ kun } 0 \leq \delta_s \leq \delta_r \leq 1. \end{cases} \tag{4.4.1}$$

Siirtomatriisilla on lievästi alipäästösuodattava vaikutus, mutta voidaan osoittaa, että määrittelemällä siirtomatriisi edellä esitetyllä tavalla minimoidaan siirron yhteydessä kuvassa tapahtuva diffuusio, toisin sanoen edellä esitetty siirto on optimaalinen kuvan alipäästösuodattumisen suhteen [4].

4.5. Lopullinen vaipan tasokuva

Edellä esitettyjen vaiheiden jälkeen venytettyjen kuvien sekvenssin perusteella voidaan koota alipikselitarkka esitys purkin vaipasta (tai etiketistä).

Lopulliseen kuvaan liitetään siis alipikselisiirretyt alkeiskuvat. Jos kahden alkeiskuvan välinen siirtymä on (dr, ds), niin alipikselisiirron suuruus (δ_r, δ_s) saadaan erotuksesta

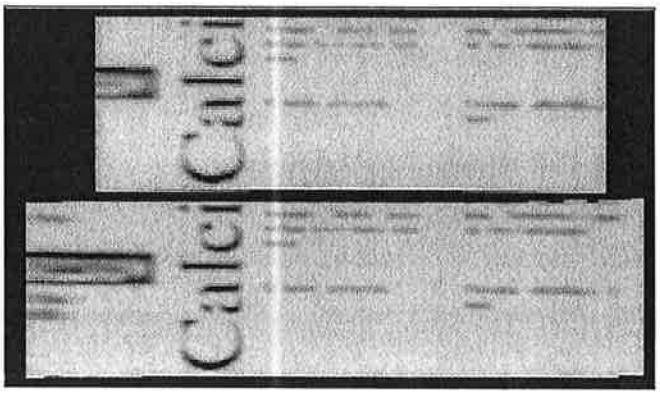
$$(\delta_r, \delta_s) = (dr, ds) - \lfloor (dr, ds) \rfloor .$$

Näin määriteltynä alipikselisiirron suuruus on aina välillä $[0, 1]$. Kun alkeiskuville on suoritettu alipikselisiirrot, käytetään yhdistämisessä rivi- ja sarakesiirtyminä alaspäin pyöristettyjä kokonaislukuja $\lfloor (dr, ds) \rfloor$.

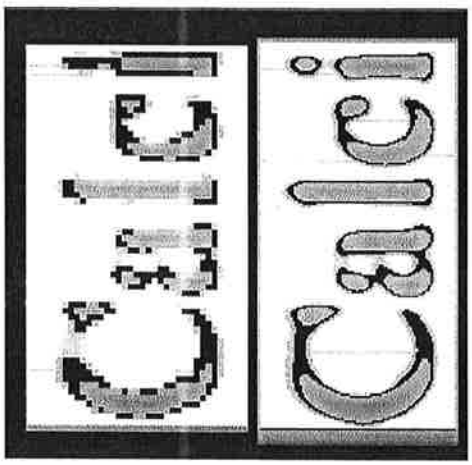
Kuva 4.5.1: Alkeiskuvien keskiarvo.

Eräs esitys purkin vaipasta saadaan, kun lasketaan alkeiskuvien keskiarvo huomioiden kullekin alkeiskuvalle sitä vastaava siirtymä (Luvussa 3.7 esitetty algoritmi). Vaihtoehtoinen tapa yhdistää alkeiskuvat yhdeksi kuvaksi, on laskea kunkin pikselin arvoksi keskiarvon sijasta päällekkäin asettuvien pikseliarvojen mediaani. Näin meneteltynä kuvassa säilyvät särmät paremmin, mikä joissakin tapauksissa on tärkeää lopullisen kuvan tarkastelussa.

Kuvassa 4.5.2 on esitetty yksittäinen alkeiskuva, sekä kahdeksan alkeiskuvan mediaani kaksinkertaisella resoluutiolla. Kuvassa 4.5.3 on osasuurennos alkeiskuvasta ja kertoimella 3 muodostetusta mediaanikuvasta (resoluutio kolminkertainen) sovitettuna samaan mittakaavaan. Kirjainten reunat on korostettu nollaamalla tietyllä harmaasävyvälillä olevat pikselit. Kuvasta nähdään, mikä vaikutus on usean eri kuvan informaatioiden yhdistämisellä verrattuna yksittäiseen kuvaan.



Kuva 4.5.2: Alkeiskuvien mediaani yhdistettynä kuvana.



Kuva 4.5.3: Osasuurennos mediaanikuvasta.

4.6. Loppusanat

Lopullisen kuvan laatuun vaikuttaa menetelmä, jolla eri kuvien sisältämä kuvainformaatio yhdistetään ja se, kuinka alkeiskuvia käsitellään ennen yhdistämistä. Kuvaan vaikuttaa myös se kuinka tarkasti alkeiskuvien välinen liike pystytään määrittämään. Lisäksi venytyksessä mahdollisesti syntyvää virhettä voitaisiin ajatella korjattavan laskemalla kuvien välinen liike yhden liikevektorin sijaan kokonaisella optisen vuon kentällä. Tällöin kuvien yhteenliittäminen alipikselisiirtoineen muuttuisi kuitenkin oleellisesti mutkikkaammaksi operaatioksi.

Onko syntynyt kuva hyvä ja mitä "hyvällä" kuvan laadulla sitten tarkoitetaankin, riippuu aina siitä, mitä lopullisesta kuvasta halutaan selvittää. Toisin sanoen yleispätevää menetelmää erilaisten kuvien paremmuusjärjestykseen laittamiseksi ei ole olemassakaan.

Kirjallisuus

[1] **FRÄNTI Pasi**, *Image Processing and Compression,* Lecture notes, Dept. of Computer Sc. University of Joensuu, 1995.

[2] **GASSON Peter C.**, *Geometry Of Spatial Forms*, Halsted Press, Chichester, painovuosi tuntematon.

[3] **NEWTON Marcus,** "REAL-TIME 3D...", *Byte vol.9,* 1984.

[4] **ROE P. L. ja SIDILKOVER D.**, "OPTIMUM POSITIVE LINEAR SCHEMES FOR ADVECTION IN TWO AND THREE DIMENSIONS", *SIAM J. Numer. Anal. vol.29,* 1992.