Norges miljø- og
biovitenskapelige
universitet

Masteroppgave 2023 30 stp
Fakultet for realfag og teknologi

Fotogrammetrisk metode for modellering av bruer: En undersøkelse av rustdeteksjon på stålbjelker

Photogrammetric method for bridge modeling:
A study of rust detection on steel beams

Hågen Paus og Jørgen Bonden
Geomatikk

Norges miljø- og
biovitenskapelige
universitet

Fakultet for realfag og teknologi

Institutt for geomatikk

Fotogrammetrisk metode for modellering av bruer: En undersøkelse av rustdeteksjon på stålbjelker

Forfattere:
Hågen Paus, Jørgen Bonden

Veiledere:
Ivar Maalen-Johansen, Institutt for geomatikk, NMBU
Tarjei Karlsen Bruaas, Statens Vegvesen

Mai, 2023

Abstract

This thesis is written on behalf of the Norwegian Public Roads Administration and investigates the possibilities of using drones as a supplement in bridge inspections. The purpose of the study is to examine whether it is possible to compose three-dimensional digital models of beam bridges using drone images and whether rust can be detected on these models.

The thesis presents a generalized flight plan that forms the basis for creating photogrammetric models of bridges. The flight plan has been tested in practice, and models of a beam bridge have been created. Three different rust detections have been performed by using RGB-values. Two of the techniques are based on manually defined threshold values, and the last is a classification done with the machine learning algorithm Random Forest.

The results show that the proposed flight plan provides good coverage of the bridge and can be used to compile a photogrammetric model. The model gives varied results on different parts of the bridge and struggles to reproduce the H-beams where they have the most homogeneous surfaces.

However, the rust surfaces are modeled well, and by using surface models it is possible to perform area calculations on rust areas. It is possible to detect rust using RGB values, but the classification itself has yielded unreliable results.

Sammendrag

Denne oppgaven er skrevet for Statens vegvesen og undersøker mulighetene for dronebruk som et supplement i bruinspeksjon. Formålet med oppgaven er å undersøke om man kan sammenstille tredimensjonale digitale modeller av bjelkebruer ved hjelp av dronebilder, og om man kan detektere rust på disse modellene.

Oppgaven legger fram en generalisert flyplan som danner grunnlaget for å lage fotogrammetriske modeller av bruer. Flyplanen har blitt testet i praksis, og det har blitt laget modeller av en bjelkebru. Det har blitt gjort tre forskjellige rust-deteksjoner ved hjelp av RGB-verdier. To av teknikkene er basert på manuelt definerte terskelverdier. Den siste er en klassifisering med maskinlæringsalgoritmen Random Forest.

Resultatene viser at flyplanen som har blitt foreslått gir en god dekningsgrad av brua, og kan bli brukt til å sammenstille en fotogrammetrisk modell. Modellen gir varierende resultater på ulike deler av brua, og sliter blant annet med å gjengi H-bjelkene der de har mest homogene flater.

Rustflatene blir derimot modellert bra, og ved bruk av overflatemodell er det mulig å gjøre arealberegninger på rustflater. Det er mulig å detektere rust ved hjelp av RGB-verdier, men klassifiseringen har gitt upålitelige resultater.

Forord

Denne masteroppgaven markerer slutten på våre fem år ved Norges miljø- og biovitenskapelige universitet.

Vi ønsker å takke vår hovedveileder Ivar Maalen-Johansen for interessen vist for oppgaven, samt alle de gode innspillene og raske tilbakemeldingene.

Takk til veileder Tarjei Bruaas i Statens vegvesen og Marius Andersen fra SINTEF for god oppfølging og tilrettelegging underveis, samt en takk til resten av de involverte i prosjektet. Takk til Jenny Bonden, Sigrid Fause Lian og Hans Kristian Bonden for korrekturlesning og tilbakemelding.

Vi vil til slutt takke medstudentene her på NMBU for fem uforglemmelige år som studenter. Vi er takknemlige for mange av de livslange vennskapene stiftet gjennom klassemiljø, linjeforening og Studentsamfunnet. Takk til gjengen på mastersalen for muntert selskap gjennom hele masterperioden.

Innhold

1 Innledning **1**

1.1 Bakgrunn . . . 1
1.2 Problemstilling og formål . . . 2
1.3 Oppbygging . . . 2

2 Teori **3**

2.1 Fotogrammetri . . . 3
2.1.1 Det perspektiviske sambandet . . . 3
2.1.2 Ytre orientering . . . 4
2.1.3 Indre orientering . . . 5
2.1.4 Relativ orientering . . . 6
2.1.5 Structure-from-Motion . . . 6
2.1.6 Strålebuntutjevning . . . 8
2.2 Geodetisk datum . . . 8
2.3 Kartprojeksjon . . . 9
2.4 Posisjonering . . . 9
2.4.1 RTK . . . 10
2.5 Kameraparametre . . . 11
2.5.1 Eksponeringstrekanten . . . 11
2.6 Flyplanlegging . . . 12
2.6.1 Ground Sampling Distance . . . 12
2.6.2 Overlapp . . . 14
2.6.3 Overlapp og kameravinkel for kanter . . . 15
2.6.4 Sammenheng mellom lukkertid og flyhastighet . . . 16
2.6.5 Bildefrekvens i fartsretning . . . 16
2.7 RANSAC . . . 17
2.8 Random Forest . . . 18

3 Metode **19**

3.1 Utstyr . . . 19
3.2 Feltarbeid . . . 20
3.2.1 Flygning . . . 21
3.2.2 Kontrolldata . . . 24

3.3 Intervju . . . 25
3.4 Programvare . . . 25
3.4.1 Python-moduler . . . 26
3.5 Fremgangsmåte Agisoft Metashape Professional . . . 26
3.6 Koordinatkonvertering . . . 28
3.7 Segmentering med RANSAC . . . 28
3.8 Evaluering av modellens nøyaktighet . . . 29
3.8.1 Metrisk evaluering av modell . . . 30
3.8.2 Fremgangsmåte . . . 30
3.9 Deteksjon av rust . . . 31
3.9.1 Fargekriterier ved terskelverdier . . . 31
3.9.2 Filtrering av punktsky . . . 32
3.9.3 Filtrering av overflatemodell . . . 32
3.9.4 Rustdeteksjon med Random Forest . . . 32
4 Resultater **33**
4.1 Bildekvalitet . . . 33
4.1.1 Blur i skråbilder . . . 34
4.2 Resultat modellprosessering . . . 35
4.2.1 Modeller . . . 35
4.2.2 Prosesseringstid . . . 38
4.2.3 RMSE verdier . . . 38
4.2.4 Tverrsnitt punktskyer . . . 39
4.2.5 Resultater av H-bjelker . . . 41
4.2.6 Sammenligning av tverrsnitt . . . 42
4.2.7 Kjennetegn ved de forskjellige modellene . . . 43
4.2.8 Mangler i modellene . . . 44
4.3 Laserscan . . . 46
4.4 LiDAR fra drone . . . 47
4.5 Sammenligning laserscan og fotogrammetri . . . 47
4.5.1 Punkttetthet . . . 47
4.5.2 RANSAC i pre-prosessering . . . 47
4.5.3 Sky-til-sky-distanse . . . 49
4.6 RANSAC-segmentering . . . 50
4.7 Deteksjon av rust . . . 51
4.7.1 Punktsky . . . 51
4.7.2 Fordeling av RGB-verdier i klassifisering . . . 55
4.7.3 Rustdeteksjon i overflatemodell . . . 58
4.7.4 Feilklassifisering . . . 60
5 Diskusjon **61**

5.1 Modell . . . 61
5.1.1 Overordnet inntrykk av modellene . . . 61
5.1.2 Georeferering . . . 63
5.2 Erfaringer ved praktisk gjennomføring . . . 63
5.2.1 Flyplan . . . 63
5.2.2 Bildetagning . . . 64
5.3 Deteksjon av rust . . . 64
5.3.1 Forbedringer av deteksjonene . . . 65
5.3.2 Svakheter ved deteksjon ved rust på fotogrammetrimodell . . . 65
5.4 Videre arbeid . . . 66
5.4.1 Implementering av multispektrale bilder i modellering . . . 66
5.4.2 Automatisk segmentering av bjelkene . . . 66
6 Konklusjon **67**
A Tabeller **71**
B Pythonkode **73**
B.1 Transformasjon mellom koordinatsystemer . . . 73
B.2 Rustdeteksjon Punktsky . . . 76
B.3 Rustdeteksjon Mesh . . . 78
B.4 Rustdeteksjon Random Forrest . . . 80
C Agisoft Metasape Modell 1 rapport **83**
D Agisoft Metasape Modell 2 rapport **92**
E Agisoft Metasape Modell 3 rapport **103**
F Agisoft Metasape Modell 4 rapport **114**

Figurer

2.1 Ytre orientering 5
2.2 Indre orientering 6
2.3 Structure-from-Motion 7
2.4 Eksponeringstrekanten 12
2.5 Sammenheng mellom GSD og kameraparametre 13
2.6 GSD i konkave og konvekse objekter 14
2.7 Illustrasjon av overlapp 15
2.8 Overlapp for kanter 15
2.9 RANSAC implementert på en punktmengde. 17
2.10 Beslutningstre 18

3.1 DJI Matrice 300 RTK 19
3.2 Uvesund bru 21
3.3 Illustrasjon av kameraposisjoner 22
3.4 Bildetagning under brua 22
3.5 Illustrasjon av bildetagning av den nederste delen av brua. 23
3.6 Oppsillingspunkt og GCP-er 24
3.7 Fem testflater for C2C 29
3.8 Distanse mellom to punktskyer 30

4.1 Eksempler på bilder av Uvesund bru tatt under droneflyging 33
4.2 Blur på skråbilder 34
4.3 Overflatemodellen av Uvesund bru 35
4.4 Bilde tatt med P1 36
4.5 Overflatemodell 36
4.6 Bilde av punktsky 37
4.7 Punktsky med konfidensnivå 37
4.8 Tverrsnitt fra Modell 1 39
4.9 Tverrsnitt fra Modell 2 39
4.10 Tverrsnitt fra Modell 3 40
4.11 Tverrsnitt fra Modell 4 40
4.12 Tverrsnitt ytre H-bjelker 41
4.13 Tverrsnitt midtre H-bjelker 42
4.14 Tverrsnitt av Modell 1 og Modell 2 42
4.15 Sammenligning av veibaneoverflaten i Modell 2 og 3. 43
4.16 Vanskelig parti på brua 44
4.17 Problemområde under brua 44
4.18 Hull i punktskyen på midterste H-bjelke 45

4.19 Scan fra totalstasjon på bakken 46
4.20 Fotogrammetrimodell sammenlignet med laserscan 46
4.21 Fotogrammetrimodell sammenlignet med LiDAR-scan 47
4.22 RANSAC implementert på bjelke 48
4.23 RANSAC implementert på flate fra pilar 48
4.24 Flatesegmentering med RANSAC 50
4.25 Flatesegmentering med RANSAC ved terskelverdi 0.5 50
4.26 Resultat deteksjon av rust figur 1 52
4.27 Resultat deteksjon av rust figur 2 53
4.28 Plott av RGB-fordeling 55
4.29 Fordeling av rust-punkter 56
4.30 Fordeling av rust-punkter 57
4.31 Rust på overflatemodell 58
4.32 Klassifisering av rust under brua 60

5.1 Fellespunkter i bilder 62
5.2 Flater på H-bjelkene som ikke blir dekket av flyplanen 63

Tabeller

3.1 Zenmuse H20 spesifikasjoner 20
3.2 Zenmuse P1 spesifikasjoner 20
3.3 Eksempel på GSD ved avstand 10 meter 23
3.4 Maksimal lukkertid for å unngå blur 24
3.5 Milde terskelverdier 31
3.6 Strenge terskelverdier 32

4.1 Prosesstider for de forskjellige modellene 38
4.2 Spesifikasjoner for PC brukt til prosessering 38
4.3 RMSE-verdiene for modellen med alle bildene (Modell 2) 38
4.4 RMSE-verdiene for modellen uten nadir-bildene (Modell 3) 38
4.5 Tydelige kjennetegn ved de forskjellige modellene 43
4.6 Punkttettheter 47
4.7 Sky-til-sky-distanser med Modell 4 49
4.8 Sky-til-sky-distanser med Modell 2 49
4.9 Sky-til-sky-distanser med Modell 1 49
4.10 Antall klassifiserte rustpunkter 54
4.11 Resultatene fra rustdeteksjon på overflatemodellen 59

A.1 Punkter som ble brukt ved mild klassifisering 71
A.2 Minste og største verdiene i datasettet ved de milde terskelverdiene . 71
A.3 Punkter som ble brukt ved streng klassifisering 72
A.4 Minste og største verdiene i datasettet ved de strenge terskelverdiene 72

Begreper

Fix-løsning - Avstand til hver satellitt er kjent i antall hele bølgelengder. Avhengig av fix-løsning for høy nøyaktighet i GNSS-målinger.

Periode - Tiden mellom hver gjentakelse av et fenomen som gjentar seg med jevne mellomrom.

Mesh - En digital representasjon av et objekt hvor overflaten er delt inn i små polygoner som til sammen danner en tredimensjonal struktur.

Metadata - Informasjon som beskriver annen informasjon.

LiDAR - Fjernmålingsmetode som bruker pulserende laserlys til å måle avstander og skape høyoppløselige digitale punktskyer.

Nadir-bilde - Fotografi tatt med kameraet pekende rett ned mot bakken.

Kapittel 1

Innledning

1.1 Bakgrunn

Statens vegvesen og fylkeskommunene forvalter i dag over 17 000 bruer langs norske veier. Bruer kan i mange sammenhenger beskrives som samfunnskritiske og må holdes i forskriftmessig stand. Vedlikehold av korrosjonsbeskyttende belegg på stålbruer er i dag en stor kostnad.

Denne oppgaven er underlagt prosjektet "Reduserte livsløpskostnader for bruer – beslutningsstøtteverktøy for optimalisert vedlikehold av overflatebeskyttelse", et prosjekt som gjennomføres av Statens vegvesen i samarbeid med SINTEF. Formålet med prosjektet er å utvikle et beslutningsverktøy som på sikt kan minimalisere kostnadene for vedlikehold av korrosjonsbeskyttende belegg på bruer. Dette verktøyet skal sammenstille relevante opplysninger om bruer og gi et godt vurderingsgrunnlag for når og hvor vedlikehold bør skje. Som regel er det vanskelig å gjøre beslutninger om når vedlikeholdet skal begynne. Vedlikeholdet kan enten begynne tidlig i skadeforløpet, eller vente og utnytte beleggets levetid maksimalt. Samtidig er det mange andre faktorer som har påvirkning på korrosjonsforløpet. Det gjenstår mange spørsmål om hvordan forløpet er i forskjellige brutyper, og om hvordan forskjellige eksponeringsmiljøer spiller inn. For å svare på dette kreves det store mengder historiske data for hver bru. Det overordnede formålet er derfor å kunne samle og sammenstille disse dataene for å predikere levetid og behov for vedlikehold i framtiden.

I prosjektbeskrivelsen fremstilles det som sentralt å ta i bruk ny teknologi i bruinspeksjon. Der blir bildeanalyse og dronebruk beskrevet som verktøy som bør utforskes videre. Det pekes på flere fordeler som bruk av droner kan bidra til:

- Økt sikkerhet, da det kan føre til mindre arbeid i høyden
- Reduserte kostnader ved økt effektivisering i inspeksjonsarbeidet
- Et mer nøyaktig og objektivt estimat av beleggets tilstand, slik at vedlikehold kan planlegges mer hensiktsmessig og likt på landsbasis

Et delmål for prosjektet til Statens vegvesen er å utforske om dronebruk kan gi et mer nøyaktig bilde av nåtilstand enn vanlig bruinspeksjon, og om det er mulig å kvantifisere skadet areal i prosent.

1.2 Problemstilling og formål

Denne oppgaven har som formål å undersøke om man kan sammenstille 3D-modeller av bjelkebruer ved hjelp av dronebilder, og undersøker om det er mulig å gjøre arealberegninger av korrosjon.

Problemstillingen er delt inn i to spørsmål:

1. Hvordan bør det tas bilder av en bru hvis den skal modelleres i en fotogrammetriprosess, og hva slags kvalitet kan oppnås i en slik modell?
2. Kan man klassifisere og kvantifisere korrosjon på stålbjelkene ved hjelp av en fotogrammetrimodell?

1.3 Oppbygging

Oppgaven er delt inn i seks kapitler:

1. **Innledning**
   Introduksjon til tema for oppgaven og problemstilling
2. **Teori**
   Teorien bak metodene brukt i oppgaven
3. **Metode**
   Beskriver utstyr, feltarbeid og metoder brukt i oppgaven
4. **Resultat**
   De forskjellige modellene legges fram og analyser presenteres
5. **Diskusjon**
   Resultatene og metode blir satt opp mot problemstillingen og drøftet
6. **Konklusjon**
   Oppgaven oppsummeres og problemstilling blir besvart

Kapittel 2

Teori

2.1 Fotogrammetri

Fotogrammetri er læren som omhandler måling av geometriske egenskaper i fotografiske bilder. Ved å anvende fotogrammetri kan form, størrelse og beliggenhet av objekter som er fotografert bestemmes (B. Dick, 2020a). En av metodene som brukes innen fotogrammetri er stereofotogrammetri. Dette innebærer å estimere tredimensjonale koordinater fra et bildepar. Stereofotogrammetri går ut på å finne fellespunkter i to eller flere bilder og bruke disse til å rekonstruere en siktelinje eller "stråle" mellom kameraposisjon og det tredimensjonale punktet. Ved hjelp av en trianguleringsprosess kan det beregnes hvor i rommet disse strålene fra kameraposisjonene krysser hverandre.

I denne oppgaven blir fotogrammetri brukt til å danne digitale 3D-modeller. Prinsippene for fotogrammetrisk modellering vil bli presentert videre i dette delkapittelet.

2.1.1 Det perspektiviske sambandet

Det perspektiviske sambandet beskriver forholdet mellom bildekoordinater og virkelige koordinater, gitt projeksjonssentrum og en rotasjonsmatrise mellom de to koordinatsystemene. Ligningene for det perspektiviske sambandet som skrevet i Andersen (2003):

$$x_b = -c\,\frac{a_{11}(x_\mathrm{T} - x_0) + a_{12}(y_\mathrm{T} - y_0) + a_{13}(z_\mathrm{T} - z_0)}{a_{31}(x_\mathrm{T} - x_0) + a_{32}(y_\mathrm{T} - y_0) + a_{33}(z_\mathrm{T} - z_0)} \tag{2.1}$$

$$y_b = -c\,\frac{a_{21}(x_\mathrm{T} - x_0) + a_{22}(y_\mathrm{T} - y_0) + a_{23}(z_\mathrm{T} - z_0)}{a_{31}(x_\mathrm{T} - x_0) + a_{32}(y_\mathrm{T} - y_0) + a_{33}(z_\mathrm{T} - z_0)} \tag{2.2}$$

der:

x_T, y_T, z_T = terrengkoordinatene (ofte ukjente)
x_0, y_0, z_0 = projeksjonssenterets koordinater (ofte ukjente)
a_{ij} = elementer i rotasjonsmatrisa A, funksjon av tre dreininger (ofte ukjente)
c = kamerakonstant (ofte kjent)
x_b,y_b = bildekoordinater (ofte kjente)

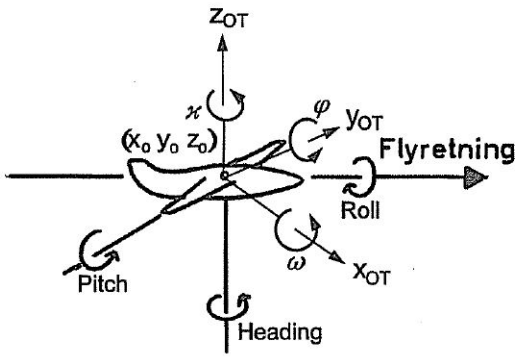
Disse ligningene gjør det mulig å regne om fra bildekoordinater til terrengkoordinater og motsatt. De forutsetter at vi har en feilfri sentralprojeksjon. Det vil si at punkter blir overført til projeksjonsplanet ved rette linjer gjennom objektivets midtpunkt (B. Dick, 2020b). Dette vil ikke være en realitet da vi alltid vil ha en forvrenging av bildet på grunn av linsen i kameraet. Det vil si at lyset ikke følger en perfekt rett linje fra objekt til projeksjonssentrum. Dette må tas hensyn til i ligningene for å få en nøyaktig modell.

Ligningene inneholder det vi kaller for indre og ytre orienteringselementer. Disse blir utdypet nærmere i 2.1.2 og 2.1.3

2.1.2 Ytre orientering

Ytre orientering refererer til strålebuntens posisjon når et bilde blir tatt. Det vil si hvor bildet ble tatt og dets pekeretning. Ytre orientering kan beskrives ved hjelp av seks parametere:

- x_0, y_0, z_0
  - Disse tre koordinatene angir projeksjonssenterets posisjon i bildetagningstidspunktet.
- ω, φ, κ
  - De tre rotasjonsvinklene til kameraet.



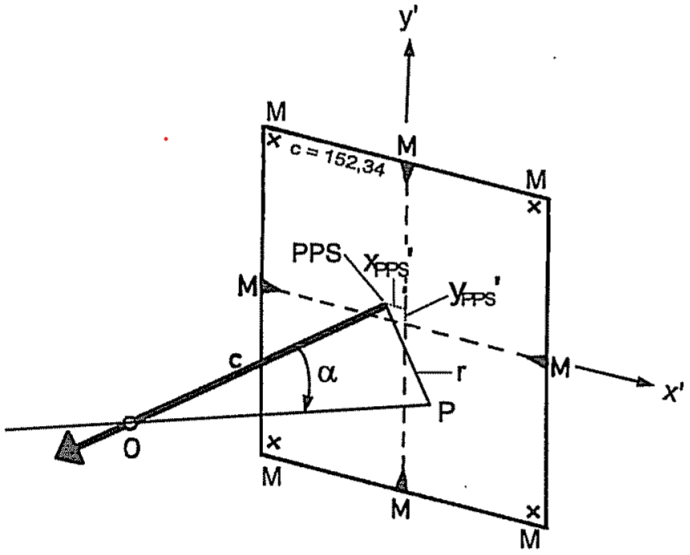
Figur 2.1: Ytre orientering. Illustrasjon hentet fra Andersen (2003).

2.1.3 Indre orientering

Indre orientering omfatter de variablene som beskriver kameraets indre geometri. Disse påvirker strålebuntens form i det bildet blir tatt (Andersen, 2003).

Parametrene for indre orientering:

- **Hovedpunktets bildekoordinater $\mathbf{x}_{PPS}$ og $\mathbf{y}_{PPS}$**
  - Punktet der normalvektoren ut fra bildeplanet vil treffe projeksjonssenteret O.
- **Kamerakonstant** c
  - Lengden på vektoren mellom projeksjonssenter og hovedpunktet.
- **Objektivets fortegning**
  - Avbøyning i objektivet. Deles inn i radiell fortegning og tangentiell fortegning. Kommer av avvik i sammenfallende akser i linseelementene.



Figur 2.2: Indre geometri i kamera. Illustrasjon hentet fra Andersen (2003).

For å få en nøyaktig modell med fotogrammetri må parametrene for den indre orienteringen bestemmes, dette blir gjort i en kamerakalibrering. I de fleste fotogrammetriprogrammer gjøres dette automatisk i prosessen, derfor er det ofte ikke nødvendig å gjøre dette på forhånd av bildetagningen.

2.1.4 Relativ orientering

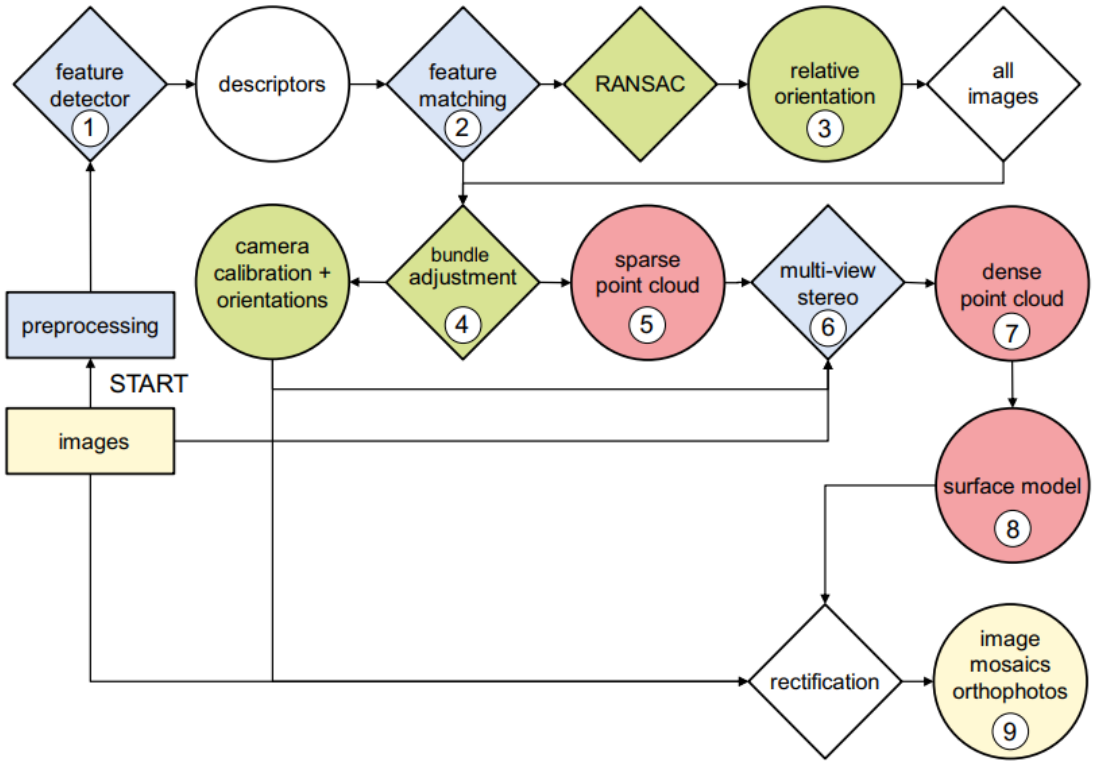
Relativ orientering beskriver posisjonen og orienteringen til bildene i forhold til hverandre (Luhmann et al., 2020). De relative orienteringene bestemmes fra sammenbindingspunkter i bildene og danner grunnlaget for videre beregninger i strålebuntutjevningen.

2.1.5 Structure-from-Motion

Structure-from-Motion (SfM) beskriver den digitale teknikken som brukes for å sammenstille 3D-modeller av bilder. Dette er prosessen moderne fotogrammetridataprogrammer tar i bruk.

SfM kan regnes som et generisk uttrykk for mange flytprosesser innenfor fotogrammetri. Prosessen består av en rekke algoritmer som til slutt ender med en tett punktsky. Produsenter av digitale fotogrammetriprogrammer bruker som regel forskjellige metoder for SfM. Mange av de beste programmene er kommersielle, derfor er det ofte skjult kildekode.

En generell flytprosess i SfM kan beskrives slik som i figur 2.3.
SfM fungerer som regel best der det er god overflatetekstur, god overlapp på bildene og et kamera som egner seg til fotogrammetri (Luhmann et al., 2020).



Figur 2.3: En generell flytprosess som ofte ses i Structure-from-Motion. Figur fra Luhmann et al. (2020).

Feature detection

"Feature detection" (FD) beskriver algoritmen/prosessen for å identifisere distinkte bildepunkter/bildeområder som er kandidater for bildematching (Luhmann et al., 2020). Algoritmen finner det som ofte kalles interessepunkt i bilder. Dette er punkter som har høy sjanse for å holde seg synlige under forskjellige lysforhold og bildevinkler.

Det finnes mange ulike algoritmer for dette formålet. En av de vanligste heter scale-invariant feature transform (SIFT). Denne algoritmen går ut på å detektere interessepunktene og deretter hente ut karakterestikkene som definerer punktet/området. Disse punktene er robuste selv ved endring av skala, rotasjon og affin transformasjon (Lowe, 2004).

Feature matching

I en "Feature matching" (FM) brukes beskrivelsene gitt i FD til å sammenligne de forskjellige punktene og deretter bestemme hvem som sammenfaller. Disse punktene danner grunnlaget for strålebuntutjevningen.

2.1.6 Strålebuntutjevning

Strålebuntutjevning er teknikken som sammenstiller bilder og estimerer tredimensjonale koordinater. I et overbestemt system av ligninger vil det estimeres koordinater for punkter i modellen, samt ytre og indre orienteringsparametre. Formålet med metoden er å finne parametrene som minimerer reprojeksjonsfeilen. Dette betyr at "strålebuntene" fra de forskjellige kameraposisjonene bør krysse hverandre i sine korresponderende objektpunkter med minst mulig avvik (Luhmann et al., 2020). Med alle de korresponderende punktene i forskjellige bilder funnet i FM kan de matematiske sammenhengene gitt i ligningene for det perspektiviske sambandet utnyttes. Metoden går ut på å minimere diverse tapsfunksjoner basert på disse sammenhengene, dermed blir alle parametrene estimert i en "bunt" sammen (Triggs et al., 2000).

2.2 Geodetisk datum

Et datum refererer til et system for måling og beregning, og brukes til å spesifisere et koordinatsystem (Skogseth og Norberg, 2014). Geodetiske datum legger grunnlaget for geografiske koordinater i kart, og kan beskrives som en matematisk modell av jordens form. Det skilles ofte mellom horisontale og vertikale datum, der horisontale datum beskriver nord/øst og vertikale datum beskriver høyder. Grunnen til dette skillet er at høyder som regel vil knyttes til tyngdefeltet.

Datum deles også inn i lokale og globale datum. Et globalt datum dekker hele jorda og gir en felles referanse for globale posisjonsbestemmelser. Et lokalt datum kan være tilpasset et bestemt område på jorden og ha en spesielt tilpasset referanseellipsoide.

Videre presenteres noen sentrale geodetiske datum for denne oppgaven.

WGS84

WGS84 (World Geodetic System) er et globalt datum som baserer seg på en ellipsoide med sentrum i jordas massesenter (Skogseth og Norberg, 2014). Datumet er standarden for posisjonsbestemmelse med GNSS, og regnes som et av de mest brukte datumene i verden. WGS84 blir jevnlig oppdatert i henhold til ITRF for å sikre best mulig nøyaktighet med GNSS-systemer.

EUREF89

EUREF89 (også kalt ETRS89) er et lokalt geodetisk datum, og er det offisielle datumet i Norge. EUREF89 er et statisk datum, det betyr at datumet følger den naturlige bevegelsen til den eurasiske kontinentalplaten. Datumet låses til platen via et sett fastmerker, følgelig vil det være et avvik mellom EUREF89 og ITRF. (Skogseth og Norberg, 2014).

NN2000

NN2000 er det offisielle høydedatumet i Norge (Skogseth og Norberg, 2014). Nullhøyden til NN2000 tilsvarer omtrent havets middelvannstand, og derfor vil en høyde i NN2000 noe forenklet beskrevet angi en høyde over havet.

2.3 Kartprojeksjon

Kartprojeksjon beskriver prosessen med å representere en tredimensjonal jordoverflate på en todimensjonal flate som et kart (Skogseth og Norberg, 2014). Ved hjelp av matematiske transformasjoner kan punkter overføres fra den krumme ellipsoiden ned på et plan. Det er umulig å representere en tredimensjonal jordoverflate nøyaktig på en todimensjonal flate, derfor vil alle kart ha en målestokkfeil.

UTM

UTM (Universal Transverse Mercator) er en kartprojeksjon som bruker liggende sylinder-projeksjon. UTM-projeksjonen deler jorden inn i 60 soner der hver sone er på 6 grader. Dette sikrer at målestokkfaktor maksimalt er på 0.9996. Dette vil gi en feil på 4 cm for en avstand på 100 meter, noe som tilsvarer 400 ppm (Skogseth og Norberg, 2014). Sonedelingen gjør at Norge blir delt inn i tre forskjellige soner. Sone 32 som er på østlandet, sone 33 på vestlandet og sone 35 i nord-Norge. Datum som EUREF89 og WGS84 er som regel avbildet i UTM-systemet.

NTM

NTM (Norsk Tranversal Mercator) er en lokal projeksjon for Norge som brukes i bygg-og anleggsbransjen. I stedet for at Norge blir delt inn i tre soner slik det gjøres i UTM, blir det delt inn i 26 soner der hver sone er på 1 grad. Dette gir en målestokkskorreksjon på 11 ppm, noe som tilsvarer 1,1 mm på 100 meter (Skogseth og Norberg, 2014).

2.4 Posisjonering

GNSS står for Global Navigation Satellite System og er en fellesbetegnelse for teknologien som gir muligheten til å bestemme posisjoner på jorda ved hjelp av satellitter. Metoden baserer seg på signaler fra satellitter som blir mottatt av en eller flere GNSS-mottakere på jorda. Når avstanden til tre eller flere satellitter er kjent er det mulig å finne posisjonen til mottakeren med triangulering. Dette er mulig da banen til satellittene er kjent med tilstrekkelig nøyaktighet.

Avstanden til en satellitt kan måles på to måter:

1. Pseudoavstandmålinger på koden
2. Ved fasemålinger på bærebølgen

Psuedoavstandsmålinger foregår ved at satellitten sender ut et signal med en kodeserie modulert på bølgen. Koden inneholder informasjon om satellittbanen, tidspunkt for utsendt signal og andre viktige parametre. Alle satellitter har ekstremt nøyaktige satellittklokker som kan registrere tidspunktet signalet blir sendt. Mottakeren på bakken har en klokke som registrerer tidspunktet når signalet blir mottatt, men denne er betraktelig mer unøyaktig. Med denne tidsdifferansen kan det regnes ut avstanden fra satellitten til mottakeren hvis korreksjoner for diverse forstyrrelser på signalbanen blir lagt på. Med fire satellitter eller flere kan x-, y-, z-koordinater og klokkefeilen til mottakeren beregnes i en utjevning (Skogseth og Norberg, 2014). Dette er den vanlige metoden brukt i håndholdte GNSS-mottakere og har en nøyaktighet på rundt 2 meter i sanntid.

Fasemålinger måler avstanden mellom satellitt og mottaker ved å måle fasen på bærebølgen. Mottakeren registrerer en avstandsendring til satellitten i form av hele bølgelengder pluss en del av en bølgelengde (Skogseth og Norberg, 2014). Det vil si at endringen i avstanden til satellitten registreres fra det tidspunktet målingen begynte. Antallet hele bølgelengder ved starttidspunktet er i utgangspunktet ukjent, men ved å måle i et visst tidsrom kan denne ukjente beregnes. Siden bølgelengden til signalet er kjent, og man vet antall bølger til satellitten, kan avstanden til satellitten beregnes. For å bruke fasemålingene er det også nødvendig å beregne klokkeavviket fra pseudomålingene (slik beskrevet i forrige avsnitt).

For å få høyere nøyaktighet på fasemålinger kan det som blir kalt differensiell GNSS brukes. Det vil si at basestasjoner i nærheten av mottakeren står i et kjentpunkt og gjør fasemålinger. Målingene i denne stasjonen kan brukes til å korrigere målingen som gjøres i mottakeren.

2.4.1 RTK

Realtids kinematisk måling (RTK) er en form for differensiell posisjonsbestemmelse som gir nøyaktighet på centimetersnivå. Metoden er basert på fasemålinger og gir korrigerte posisjoner i sanntid uten behov for etterprosessering (Teunissem og Montenbruck, 2017).

I Norge heter RTK-systemet CPOS og er drevet av Kartverket. Dette systemet består av omtrent 160 basestasjoner jevnt fordelt utover hele landet (Skogseth og Norberg, 2014). Systemet baserer seg på å lage en Virtuell basestasjon (VRS) i nærheten av GNSS-mottakeren. Dette gjøres ved å estimere en korreksjon ved hjelp av interpolering av alle basestasjonene i området (Teunissem og Montenbruck, 2017). Denne korreksjonen sendes tilbake til GNSS-mottakeren og gir en nøyaktig posisjonering, noe som kalles for fix-løsning.

2.5 Kameraparametre

Kameraparametre beskriver konfigurasjoner som påvirker hvordan et bilde blir fanget i kameraet.

ISO

ISO er en størrelse som angir hvor sensitiv en fotosensor skal være. En høy ISO-verdi vil gi større følsomhet for lys, og store ISO-verdier vil øke mengden støy i bildet. Det er ofte ønskelig å finne en gylden middelvei der ISO er høy nok til at bildet ikke blir for mørkt, og samtidig lav nok til at det ikke blir kornete (O'Connor et al., 2017).

Lukkertid

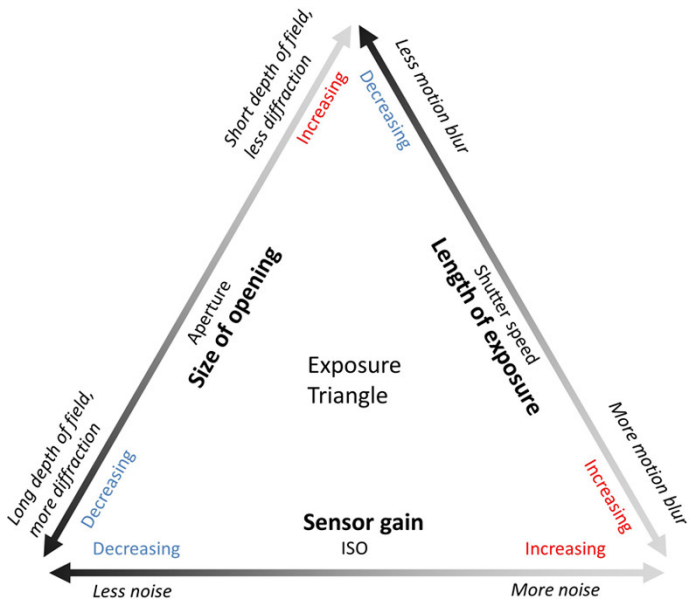
Lukkertiden bestemmer hvor lenge fotosensoren skal eksponeres for lys. En lang lukkertid vil føre til at sensoren eksponeres lengre for lys og bildet blir konsekvent lysere. Lang lukkertid vil også gjøre at bevegelser av kameraet i bildeøyeblikket i større grad vil føre til uskarpheter (O'Connor et al., 2017).

Blenderåpning

Blenderåpning er størrelsen på åpningen i linsen som lyset passerer gjennom. Blenderåpning måles i f-verdier og angir forholdet mellom diameteren på åpningen og brennvidden til linsen. Ved en lav f-verdi vil mer lys slippes inn. En lav verdi fører også til stor dybdeskarphet, noe som betyr at det blir bedre detaljer på både objektet og bakgrunnen. Det vil også føre til at bildet blir mørkere (O'Connor et al., 2017).

2.5.1 Eksponeringstrekanten

Eksponeringstrekanten (figur 2.4) beskriver hvordan forskjellige verdier for ISO, lukkertid og blenderåpning påvirker bildekvaliteten.



Figur 2.4: Eksponeringstrekanten. Figur hentet fra O'Connor et al. (2017).

2.6 Flyplanlegging

For å lage en punktsky som tilfredsstiller kravene til et prosjekt er det nødvendig med en tilrettelagt flyplan. Vanligvis skilles det mellom to forskjellige metoder for å planlegge flygninger med fotogrammetriske formål: 2D-planlegging og 3D-planlegging. 2D-planleggingen er ofte den enkleste måten. I de fleste programvarer kan et polygon som inneholder interesseområdet ditt markeres, og med et par brukerdefinerte parametre blir flyplanen automatisk generert. Denne metoden fungerer bra for flate områder, men er lite anvendelig for objekter og områder med kompleks geometri.

I dette delkapittelet blir prinsipper rundt flyplanlegging med formål for fotogrammetriske operasjoner på bruer beskrevet. Bruer er komplekse strukturer som varier i form og skala, og dette gjør at det er flere prinsipper som må implementeres i en flyplan.

2.6.1 Ground Sampling Distance

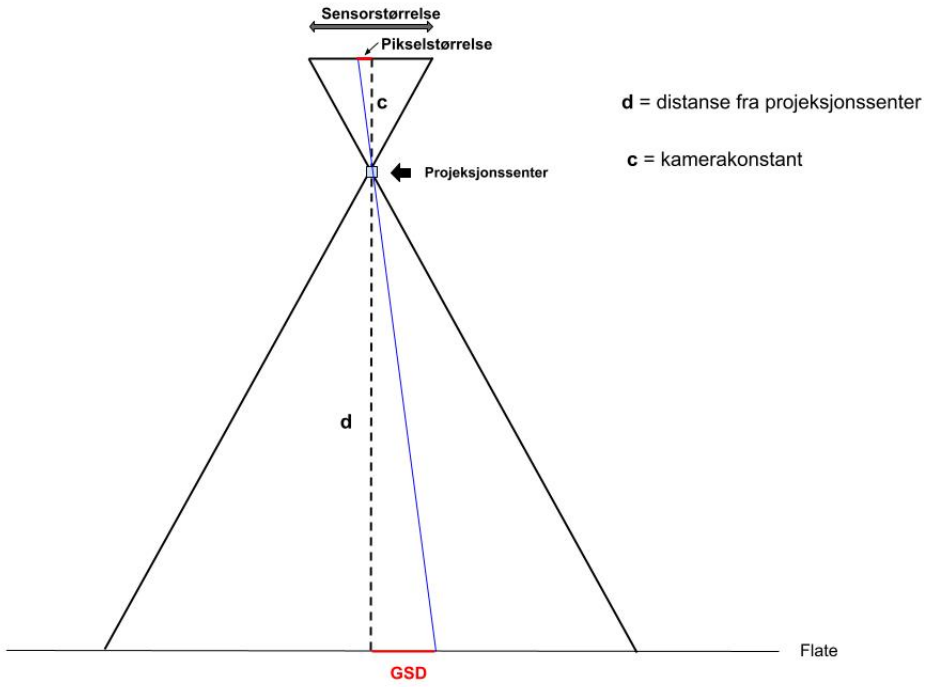
Ground Sampling Distance (GSD) er distansen en piksel i et bilde tilsvarer på en flate i virkeligheten (O'Connor et al., 2017). En lav GSD tilsvarer større romlig oppløsning, noe som er ønskelig i fotogrammetriske prosesser. For å regne ut GSD er det mulig å utnytte forholdet mellom kamerakonstant og pikselstørrelsen i bildesensoren. Dette forholdet tilsvarer forholdet mellom avstand til objekt og GSD. Dette gir oss ligning 2.3.

$$\mathrm{GSD} = \frac{\mathrm{d} \times p}{\mathrm{c}} \tag{2.3}$$

der:

d = avstand til objekt som avbildes
c = kamerakonstant
p = pikselstørrelse i bildesensor

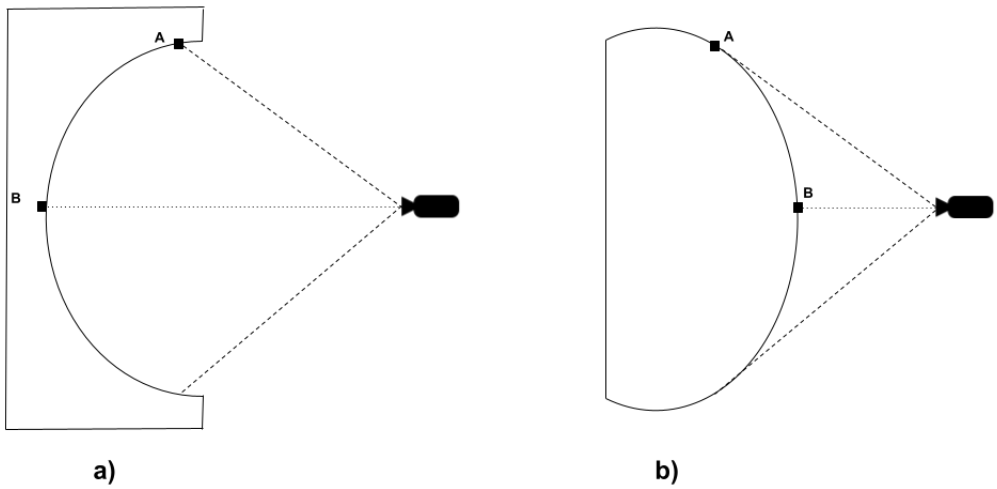
Denne ligningen gjelder når bildestrålen blir projisert vinkelrett på en flate. Ved en flate som står skrått på bildestrålen vil ikke dette være helt nøyaktig, men for en generell flyplanlegging vil dette holde. Dette illustreres i figur 2.6.



Figur 2.5: Sammenhengen mellom GSD og kameraparametre slik som beskrevet i ligning 2.3.

Med en enkel omforming av ligning 2.3 kan den løses for avstand, gitt at det settes en ønsket GSD-verdi. I sammenheng med bruinspeksjoner kan det være fornuftig å sette krav til GSD på halvparten av bredden til den defekten du vil måle (Wang et al., 2022). Skal for eksempel sprekker i betong tas bilder av, bør GSD-en være halvparten av sprekkens bredde.

Ligning 2.3 og figur 2.5 gjelder som nevnt der kameraet er orientert vinkelrett på en flate. På ujevne flater kan GSD-en variere selv om kameraet står vinkelrett på flaten. På en konveks flate vil GSD-en være lavere i midten av bildet, og motsatt på en konkav flate (Wang et al., 2022). Det samme prinsippet vil gjelde på en kant eller et hjørne. Figur 2.6 illustrerer effekten på GSD i ulike flater.



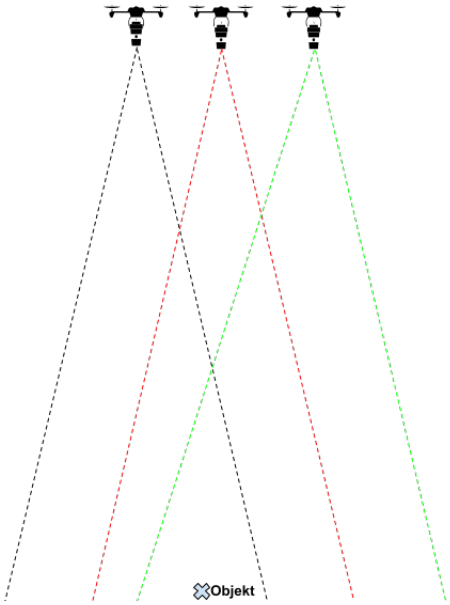
Figur 2.6: I den konkave flaten (figur a) vil punktet A ha størst GSD av de to punktene, og i den konvekse flaten (figur b) vil punktet A ha lavest GSD.

2.6.2 Overlapp

For at bildene skal være egnet til å danne en 3D-modell er det viktig med tilstrekkelig overlapp på bildene. God overlapp gir flere sammenbindingspunkter i bildematchingen, og er viktig for å få en tett nok punktsky.

Et viktig aspekt er at flyplanleggingen skal legge grunnlag for et effektivt arbeid for dronepiloten. Derfor vil det være viktig å finne en overlapp mellom bildene som gjør at modellen får tilfredstillende punkttetthet, samtidig som det ikke blir for mange bilder. Mange bilder vil øke prosesseringstiden betraktelig. Med lange bruer kan datamengdene bli veldig store, derfor vil det lønne seg å vurdere en overlapp som ikke er for stor.

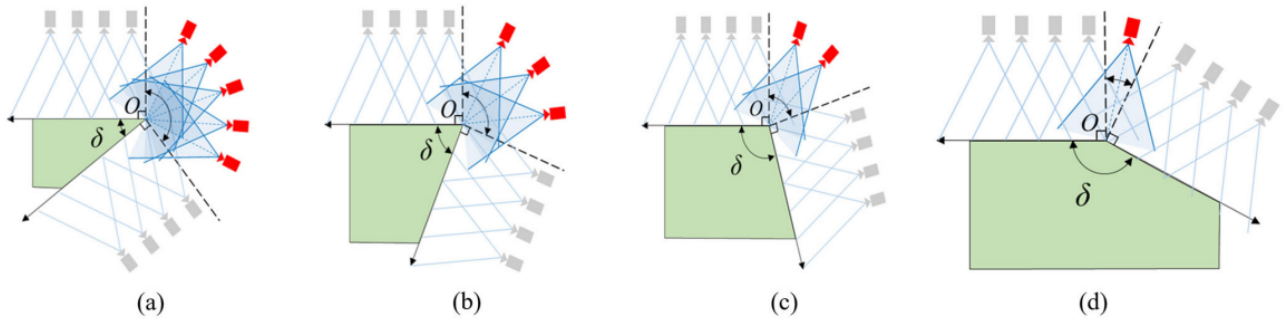
Litteraturen foreslår litt forskjellige verdier for overlapp. Nødvendig overlapp vil også variere, da forskjellig geometri blant bruer kan kreve forskjellig overlapp. Pan et al. (2019) foreslår en overlapp på 90% forover og 60% sidelengs. Wang et al. (2022) tar som utgangspunkt at et objekt i bildet bør tas bilde av fra minst tre forskjellige vinkler. Dermed blir det foreslått en overlapp på 66,7% både forover og sidelengs.



Figur 2.7: Illustrasjon av 66,7% overlapp på en flat struktur, objektet blir da med i tre bilder.

2.6.3 Overlapp og kameravinkel for kanter

På kanter er det viktig å få tilstrekkelig overlapp slik at bildematching-algoritmen klarer å finne nok fellespunkter. Antall nødvendige kameraposisjoner her vil komme an på vinkelen som dannes mellom flatene som krysser hverandre. Figur 2.8 viser hvordan vinkel $\boldsymbol{\delta}$ definerer antall kameraposisjoner:



Figur 2.8: Overlapp for kanter. Figur hentet fra Wang et al. (2022, s. 7).

(a) = $0° < \boldsymbol{\delta} < 60°$
(b) = $60° < \boldsymbol{\delta} < 90°$
(c) = $90° < \boldsymbol{\delta} < 120°$
(d) = $120° < \boldsymbol{\delta} < 360°$

2.6.4 Sammenheng mellom lukkertid og flyhastighet

Kameraparametrene som velges for eksponering vil ha mye å si for den endelige modellen, og på generell basis vil det være hensiktsmessig å justere kameraparametrene manuelt under flygningen. Autofokus kan gi store variasjoner mellom bilder og kan gi negative utslag i bildematchingen.

Ved lang lukkerhastighet og bevegelse i dronen vil det bli utydelighet i bildet som blir kalt "blur". En kvantifisering av blur kan beskrives slik som skrevet i O'Connor et al. (2017):

$$b = \frac{v \times \tau}{\text{GSD}} \tag{2.4}$$

der:

b = blur
v = fart
τ = lukkertid

En verdi på over 1,5 i ligning 2.4 vil gi blur (O'Connor et al., 2017). Det kan være lurt å holde seg godt under denne grensen, en god regel er derfor at kameraet ikke bør flytte seg en distanse som er lengre enn GSD i løpet av lukkertiden.

Lukkerhastigheten er en viktig parameter å bestemme. I forbindelse med bilder av bruer er det viktig å ha en lukkerhastighet som gir nok lys i områder med skygger. Det gjelder derfor å finne en balanse mellom avstand og fart som tillater en effektiv datainnsamling, samtidig som det unngås blur i bildene.

2.6.5 Bildefrekvens i fartsretning

Bildefrekvens vil bestemmes av ønsket overlapp, farten til dronen og avstand til objektet. Avstanden s mellom bildetagningsposisjonene:

$$s = \text{dekningsbredde} \times \text{overlapp} \tag{2.5}$$

Overlapp vil være et tall mellom 0 og 1.

Dette kan settes inn i den enkle ligningen for fart, tid og strekning ($s = v \cdot t$). Deretter løses den med hensyn på t. Dette gir perioden mellom bildetagningsøyeblikkene, altså sekundene det går mellom hvert bilde.

$$\text{bildeperiode} = \frac{\text{dekningsbredde} \times \text{overlapp}}{v} \tag{2.6}$$

Dekningsbredden kan finnes ved å utnytte at forholdet mellom kamerakonstant og sensorstørrelse tilsvarer forholdet mellom avstand til objekt og dekningsområde:

$$\text{dekningsbredde} = \text{distanse} \times \frac{\text{sensorbredde}}{\text{kamerakonstant}} \tag{2.7}$$

Det er viktig å notere at det brukes sensorbredde. Dette gjelder hvis kameraet beveger seg i bredderetningen til kameraet (sidelengs). Hvis kameraet flytter seg i lengderetningen (rett fram) brukes sensorlengde som variabel.

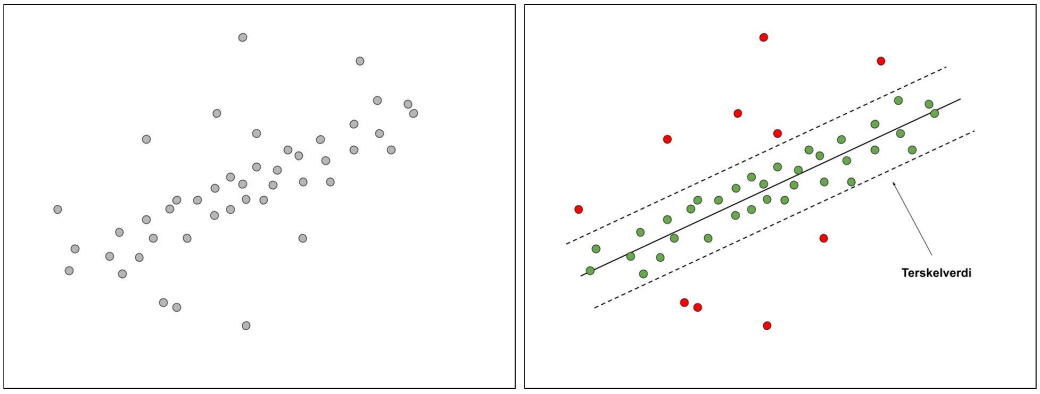
Videre settes sistnevnte uttrykk inn i ligning for bildefrekvens:

$$\text{bildeperiode} = \frac{\text{distanse} \times \frac{\text{sensorbredde}}{\text{kamerakonstant}} \times \text{overlapp}}{v} \tag{2.8}$$

Bildefrekvensen vil være inversen til bildeperioden, altså 1\bildeperiode.

2.7 RANSAC

RANSAC (Random Sample Consensus) er en metode for å estimere modeller i datasett som inneholder støyverdier. Algoritmen går ut på å velge en tilfeldig delmengde av datapunkter og estimere en modell ved hjelp av disse punktene. Modellen kan være av forskjellige typer, for eksempel en linje, sirkel eller et plan. Deretter beregnes antall datapunkter som er innenfor en bestemt lengde fra modellen. Denne lengden kalles terskelverdi. Hele denne prosessen gjentas et forhåndsbestemt antall ganger, og til slutt velges den modellen som har flest datapunkter innenfor terskelverdien til å være endelig modell (Luhmann et al., 2020).



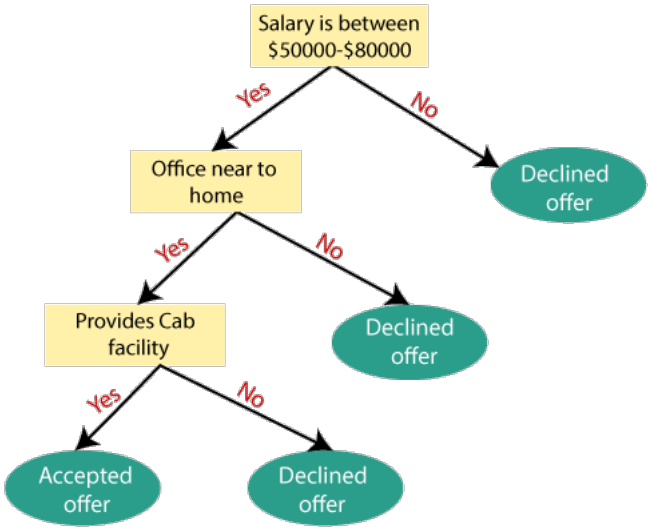
(a) Punkter i et datasett

(b) Linje tilpasset med RANSAC, der terskelverdier er illustrert med stiplede linjer

Figur 2.9: RANSAC implementert på en punktmengde.

2.8 Random Forest

Random Forest er en maskinlæringsmetode som baserer seg på å kombinere flere beslutningstre, og deretter bruke en flertallsavgjørelse for å utføre en endelig klassifisering. Et beslutningstre går ut på å dele et datasett inn i en tre-lignende struktur, der hver node representerer et sett med kriterier.



Figur 2.10: Eksempel på et beslutningstre. De gule boksene representerer noder, og de grønne ellipsene løvnoder.

Et beslutningstre bygges opp ved å bruke treningsdata for å definere noder. I hver node velges den splitten som gir størst informasjonsgevinst. Dette gjøres helt til løvnodene er "rene", noe som vil si at hvert treningobjekt i en node tilhører samme klasse. Dette er en metode som lett kan føre til overtilpasning på treningsdatasettet.

Random Forest bruker tilfeldige utvalg fra datasettet, og trener deretter mange beslutningstrær på disse. Dette fører til høy varians i klassifiseringene i de forskjellige trærne, men ved å ta majoritetsbeslutninger basert på alle trærne vil det resultere i en en robust modell som er mindre utsatt for overtilpasning (Raschka og Mirjalili, 2019).

Kapittel 3

Metode

3.1 Utstyr

Matrice 300 RTK

Matrice 300 RTK er en drone produsert av DJI, og er et av de mer avanserte dronesystemene på det kommersielle markedet. Dronen har innebygd RTK og har derfor posisjoneringsnøyaktighet på centimeter-nivå. Dette gir mulighet for å lage georefererte 3D-modeller uten å bruke GCP-er. Dronen er kompatibel med flere kameramodeller, i tillegg til LiDAR. I dette forsøket ble kameraene Zenmuse H20 og Zenmuse P1 brukt, samt Zenmuse L1 LiDAR.

Figur 3.1: DJI Matrice 300 RTK

Zenmuse H20

Zenmuse H20 er den lettere av de to kameraene. Denne modellen har mulighet for toppmontering, noe som gjør at dette kameraet egner seg til å ta bilder under bruer. Dette kameraet tar også to bilder samtidig, der et bilde blir tatt med vidvinkel og det andre med zoom. Spesifikasjoner gitt i tabell 3.1.

Tabell 3.1: Zenmuse H20 spesifikasjoner

Zenmuse H20 spesifikasjoner		
	Vidvinkel kamera	**Zoom kamera**
Sensorstørrelse	1/2.3"CMOS, 12 MP	1/1.7"CMOS, 20 MP
Fokallengde	4.5 mm	6.83-119.94 mm
Bildestørrelse (piksler)	4056 x 3040	5184 × 3888
Lukkerhastighet	1 ∼1/8000	1 ∼1/8000 s
Blenderåpning	f/2.8	f/2.8-f/11

Zenmuse P1

Zenmuse P1 har betraktelig høyere oppløsning enn H20. Kameraet har ikke mulighet for toppmontering, og derfor ble det tatt i bruk til bilder ovenifra, fra siden og mildt skrått oppover. Spesifikasjoner gitt i tabell 3.2.

Tabell 3.2: Zenmuse P1 spesifikasjoner

Zenmuse P1 spesifikasjoner	
Sensor	1/1.7"CMOS, 45 MP, 35,9x24 mm
Fokallengde	35 mm
Bildestørrelse (piksler)	8192 x 5460
Lukkerhastighet	1 ∼1/8000
Blenderåpning	f/2.8-f/16

Zenmuse L1

Zenmuse L1 er en gimbalstabilisert LiDAR som kan ta opp til 240.000 punkter per sekund og opp til tre returer. Den har også integrert en 20MP RGB-kamera med 1" CMOS sensor.

Leica MS60

Leica MS60 er en totalstasjon med innebygd scannefunksjon. I denne oppgaven ble totalstasjonen brukt til å scanne brua.

3.2 Feltarbeid

Feltarbeidet ble utført 14. og 15. februar 2023 på Uvesund bru (Nes kommune). Arbeidet ble gjort i regi av Statens vegvesen, som stilte med to dronepiloter. Uvesund bru er en 97 meter lang bjelkebru med tre stålbjelker i langsgående retning. Brua ble valgt som testobjekt på grunn av beliggenhet, størrelse og mange synlige rustflater.

Figur 3.2: Uvesund bru

Feltdagene ble planlagt etter værforhold. Ved å se på tidligere prosjekter til vegvesenet ble det konkludert med at overskyet vær var det mest ideelle for bildetagning av bruer. Begrunnelsen er at overskyet vær gir jevn belysning på hele brua, i motsetning til sollys som vil lage skygger og refleksjon i blanke overflater. Hypotesen var at dette er mindre ideelt for bildematching og detaljnivået på bildene. Snø og is under og rundt brua, kombinert med at stålbjelkene på brua er hvite, ga gode lysforhold under brua.

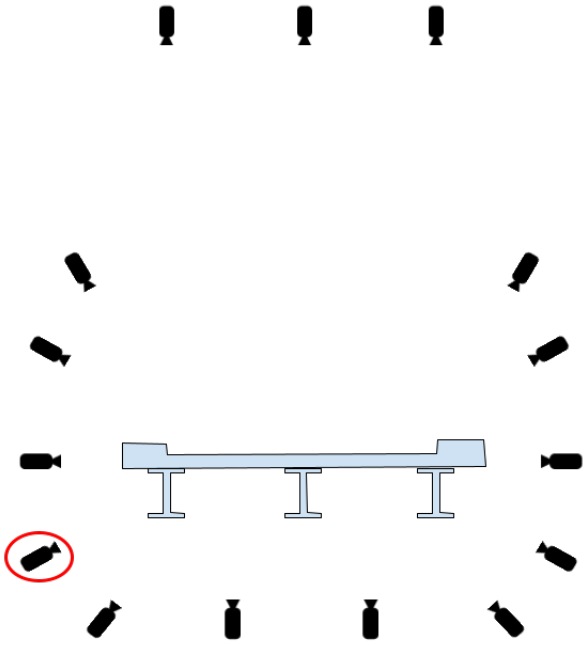
Det ble tatt totalt 3626 bilder i løpet av de to dagene.

3.2.1 Flygning

Grunnet manglende programvare for automatisk planlegging av denne type flygninger ble mesteparten av flygningen gjort manuelt. Den eneste bildeserien som ble tatt med automatisk flyplan var bildene fra oversiden av brua. Disse ble tatt 80 meter over brua, slik at det ikke var fare for kollisjon med objekter.

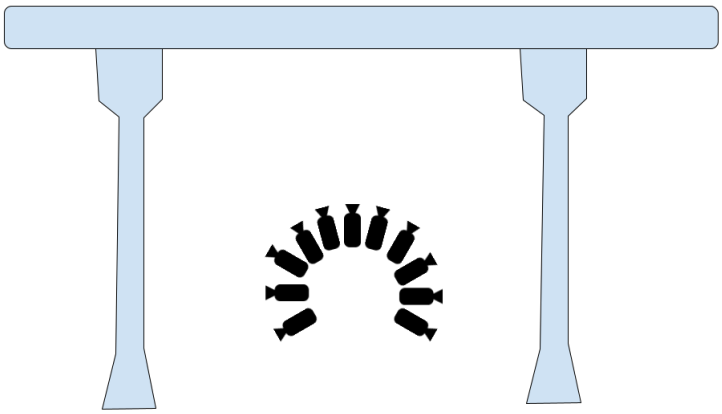
En visuell inspeksjon ga en god indikasjon på hvorfor det er komplisert å lage en automatisk flyplan for et slikt område. Vanskelige faktorer er for eksempel trær, terreng, nødvendighet for visuell kontakt med dronen, GNSS fix, høyde under brua og sikkerhetshensyn i forhold til biler.

I den manuelle flygningen ble det forsøkt etter beste evne å fly med kameraet i bestemte vinkler langs brua. Kameravinklene er illustrert i figur 3.3. De tre øverste kameraposisjonene ble som nevnt tatt med en automatisk flyplan.



Figur 3.3: Illustrasjon av omtrentlige kameraposisjoner i droneflygningen. Kamera med rød sirkel ble forsøkt vinklet skrått i flyretningen.

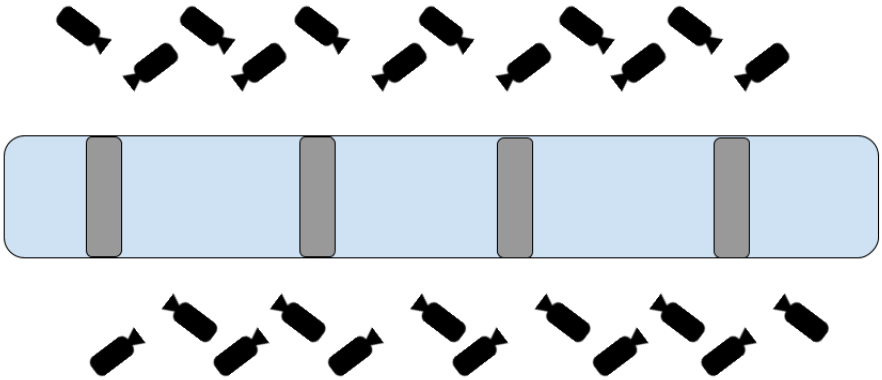
Det ble også tatt bilder fra under brua. Disse ble tatt fra så lavt som mulig slik at dronen fikk GNSS-fix. Det ble tatt bilder fra to posisjoner under brua, der hver posisjon var i midten av hver "fil" mellom stålbjelkene slik vist i figur 3.3. Fra disse posisjonene ble bildeseriene tatt i en buebevegelse, slik vist i figur 3.4.



Figur 3.4: Illustrasjon av bildetagning i buebevegelse under brua.

For å best mulig fange detaljene på betongpilarene ble det tatt skråbilder av disse. Figur 3.5 viser hvordan disse ble tatt. Det ble tatt flere bilder langs brua enn det

illustrasjonen viser.



Figur 3.5: Illustrasjon av bildetagning av den nederste delen av brua.

Dronepiloten styrte farten og retningen til dronen manuelt, samtidig som bildene ble tatt automatisk med satt frekvens. Piloten forsøkte å holde en fart på 1 m/s eller under, men det ble noen avvik fra denne farten. Det ble forsøkt å holde konstant avstand til brua under bildetagningen. På forhånd ble det sett ut en ideell avstand til brua, men i felt viste det seg å være vanskelig å holde den satte avstanden. Dette skyldtes faktorer som trafikksikkerhet og naturlige hindringer. Samtidig var det vanskelig for dronepiloten som fløy manuelt å holde styr på avstand samtidig som det ble tatt bilder. Det ble derfor noe varierende avstand til brua, men som regel oversteg den ikke 10 meter.

Tabell 3.3: Eksempel på GSD ved avstand 10 meter. Regnet ut med ligning 2.3.

Kamera	**H20 Vidvinkel**	**H20 Zoom**	**P1**
GSD	3.4 mm	2.1 mm	1.3 mm

Bildefrekvensen ble som regel holdt til 1 bilde/sekund. Dette ga overlapp på over 90 % i de fleste flyrutene. ISO og lukkertid ble justert underveis for å få mest mulig detaljer i bildet. Lyset rundt brua varierte mye, og derfor var det viktig å justere disse kameraparametrene slik at bildene ikke ble for lyse eller for mørke. ISO ble holdt så lav som mulig, mens lukkertid ble tilpasset for å unngå blur. Når lukkerhastigheten økte ble det også forsøkt å minske farten ytterligere på dronen.

Tabell 3.4 viser den maksimale lukkertiden i sekunder hvis kameraet P1 skal unngå blur.

Tabell 3.4: Maksimal lukkertid i sekunder for å unngå blur. Denne tabellen gjelder for kameraet P1, og er regnet ut med ligning 2.4.

Fart (m/s) / Avstand (m)	**0,5**	**0,75**	**1**	**1,25**
5	1/799	1/1198	1/1597	1/1997
10	1/399	1/599	1/799	1/998
15	1/266	1/399	1/532	1/666
20	1/200	1/299	1/399	1/499

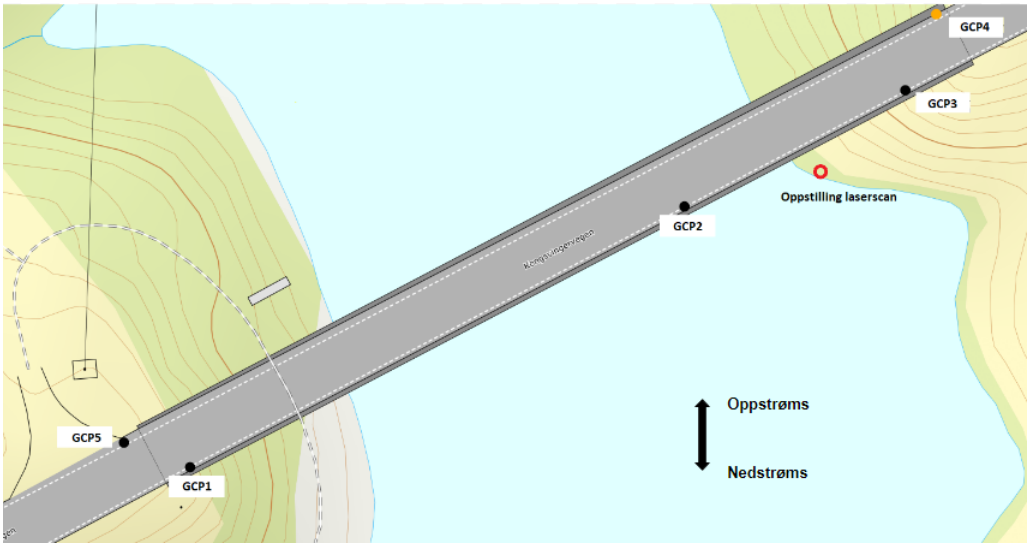
3.2.2 Kontrolldata

For å ha en sammenligning med den endelige punktskyen, ble det utført tilleggsmålinger:

GCP-er: I forkant av flygningen ble det målt inn fem kontrollpunkter (GCP-er) i området, disse ble målt inn med RTK-rover. Da dronen har innebygd RTK er ikke dette kritisk, men ble gjort for å kunne brukes til kontroll etter prosesseringen. Den fjerde GCP-en fikk slaps over seg og ble ikke synlig på bildene.

Bakkebasert laserscan: For å få et par referanseflater ble det utført et laserscan fra under brua. Det ble gjort en oppstilling med laserscanner i punktet merket rødt i figur 3.6.

LiDAR: I tillegg til bilder tatt av Zenmuse P1 og H20 ble det fanget LiDAR-data fra luften med Zenmuse L1. For å scanne krevdes det en kalibrering hvert 100 sekund, noe som gjorde det tidkrevende og vanskelig å scanne hele brua.



Figur 3.6: Oppstillingspunkt med scanner, samt alle GCP-er som ble målt inn. GCP4 markert i oransje ble ikke synlig på bildene grunnet slaps, og ble derfor ikke brukt.

Grunnet begrenset tid i feltarbeidet ble det bare gjort én oppstilling for scanning.

3.3 Intervju

I forbindelse med oppgaven ble det foretatt et intervju av en bruinspektør i Statens vegvesen. Dette var for å gi et bredere bilde av hvordan bruinspeksjon utføres i dag.

3.4 Programvare

Agisoft Metashape

Agisoft Metashape er et program som bruker fotogrammetriske prosesser for å danne 3D-modeller. I denne oppgave ble Agisoft Metashape Professional brukt, da denne produserer georefererte modeller og punktskyer. I kapittel 3.5 blir fremgangmåten i programmet gjennomgått.

CloudCompare

CloudCompare er et program som brukes til å visualisere og behandle punktskyer og annen tredimensjonal data. Programmet er basert på åpen kildekode og kan håndtere en rekke filformater, noe som gjør det anvendelig til sammenligning av modeller.

DJI Terra

DJI Terra er programvare laget av DJI som brukes til flyplanlegging og prosessering av dronedata. I denne oppgaven ble DJI Terra brukt til å konvertere rådata fra L1-LiDAR til punktsky.

Quick Terrain Modeler

Quick Terrain Modeler er programvare utviklet av Applied Imagery for 3D-visualisering og analyse av punktskyer. Programmet er egnet til å behandle store punktskyer og har verktøy for å visualisere flere punktskyer samtidig. I denne oppgaven ble programmet brukt til å produsere tverrsnitt.

Python

Python er et objektorientert kodespråk som har et bredt utvalg av moduler.

3.4.1 Python-moduler

Open3D

Open3D er en modul som gjør det mulig å behandle punktskyer og overflatemodeller i Python. Denne har blitt brukt til segmentering og rustdeteksjon på både punktsky og overflatemodell.

Laspy

Laspy er en modul som gjør det mulig å lese, redigere og produsere LAS- eller LAZ-filer i Python.

PyProj

PyProj er et bibliotek som brukes til å utføre transformasjoner mellom koordinatsystemer. I denne oppgaven ble det brukt til å transformere punktskyen fra L1-LiDAR til riktig koordinatsystem.

Scikit-learn

Scikit-learn er en modul som inneholder et utvalg av maskinlæringsalgoritmer. Denne modulen ble brukt i klassifiseringen av rustpunkter.

3.5 Fremgangsmåte Agisoft Metashape Professional

Importering av bilder

Prosessen startet ved at bildene ble importert inn i programmet. Bildene inneholdt fullstendig metadata med informasjon om blant annet posisjon og kameraparametre. Denne informasjonen ble brukt i fotogrammetriprosessen.

Allign photos

Etter at alle bildene ble importert, ble prosessen "Allign Photos" brukt. I denne prosessen ble bildematching brukt for å finne sammenbindingspunkter mellom bilder. Disse ble brukt videre til å beregne relative posisjoner og orienteringer. Her ble også kameraene kalibrert i en auto-kalibrering. I denne delen av prosessen kan parametre justeres etter behov. Nøyaktighet ble satt til høy, og kildeposisjonen satt til "source", da koordinatene i metadataen til bildene har tilstrekkelig nøyaktighet.

Under avanserte parametre ble "Key point limit" satt til 40.000 og "tie point limit" satt til 10.000.

Ground Control Point - GCP

Etter kamera-justeringen ble koordinatene til GCP-ene importert som KOF-fil. Disse ble deretter manuelt markert i bildene.

Optimalisering

I denne delen ble den indre orienteringen til kameraene optimalisert. Til dette ble prosessen "Optimize Camera Alignment" brukt. I denne prosedyren kan det velges hvilke parametre som skal bli optimalisert, og i dette forsøket ble alle parametrene optimalisert. Under avanserte parametre ble det huket av for å estimere kovariansverdier og "Fit additional corrections" for analyse av knutepunktene.

Lag punktsky

Når alt var optimalisert ble en punktsky produsert. I prosessen "Build Point Cloud" ble kvaliteten satt til høy, og kildedata satt til dybdekartene. Under avanserte parametre ble "depth filtering" satt på mild.

Lag overflatemodell

Prosessen "Lag overflatemodell" ble brukt for å lage mesh av brua. Dette ble gjort ved å bruke dybdekart som kildedata, overflatetype som "Arbitrary (3D)", kvalitet på høy, og "Face count" på høy, siden god tekstur på modellen var ønskelig ved rustdeteksjon.

Bygg tekstur

Prosessen "bygg tekstur" ble brukt for å få bedre tekstur på overflatemodellen. Her var det viktigste at teksturtypen ble satt til "Diffuse map" og kildedata ble satt til bilder.

Eksportering

Det siste steget i prosessen var å eksportere modellene. Alle punktskyene ble eksportert i las-filer, og overflatemodellene ble eksportert i obj-filer. Opprinnelig ble posisjonene på bildene registrert i WGS84, men i Agisoft Professional er det mulig å transformere koordinatsystem. Alle punktskyene ble derfor eksportert i datumet EUREF89. Koordinatsystem ble satt til UTM sone 32 med høydemodell NN2000.

Overflatemodellene ble eksportert i lokalt koordinatsystem på grunn av begrensinger i filstrukturen.

3.6 Koordinatkonvertering

Punktskyen fra drone-LiDAR kunne bare eksporteres i WGS84 i DJI Terra, derfor var det nødvendig å transformere det til samme koordinatsystem som modellene fra Agisoft. Dette ble gjort med en Python-kode som brukte PyProj-modulen.

Koden ligger i vedlegg B.1.

3.7 Segmentering med RANSAC

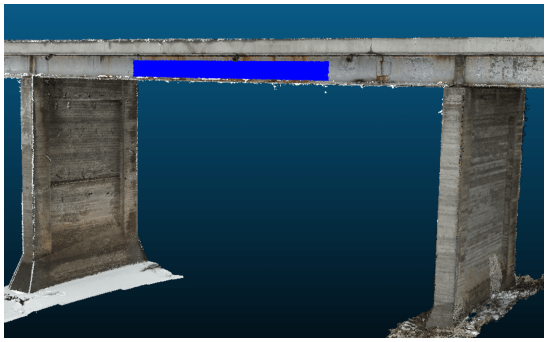
RANSAC ble brukt til segmentering av punktskyer. Til dette ble Python sin modul Open3D implementert i et koderammeverk.

Koden fungerer på følgende måte:
RANSAC-algoritmen implementeres på en punktsky, hvor den finner en flate ut i fra terskelverdi-parameteren som blir bestemt. Når flaten er funnet blir punktene som tilhører flaten fjernet fra datasettet og algoritmen kjøres på nytt på resterende punkter. Algoritmen vil iterere helt til det er en viss prosentandel igjen av datasettet som ikke er klassifisert i en flate. Denne prosentandelen er parameteren "minimumsforhold". Terskelverdi er beskrevet i 2.7.

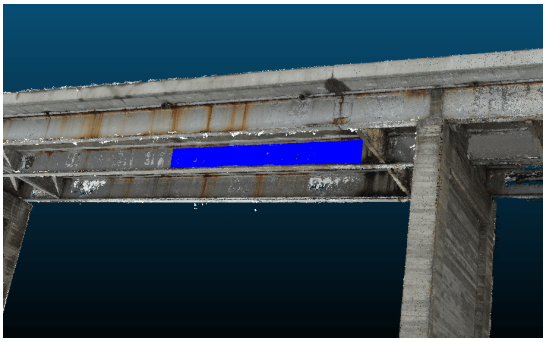
3.8 Evaluering av modellens nøyaktighet

Punktskyen som ble laget ved fotogrammetri ble validert ved hjelp av punktskyen generert av laserscanning. For å gjøre dette ble fem vilkårlige flater valgt ut og sammenlignet. Punktskyen fra laserscanningen ble altså brukt som "fasit", da punktene generert i scannet antas å ha høy nøyaktighet.

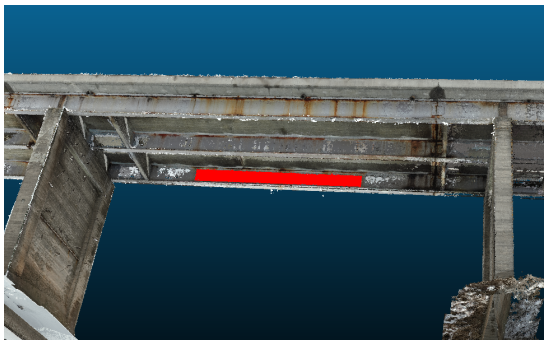
Flater som ble valgt vises i figur 3.7



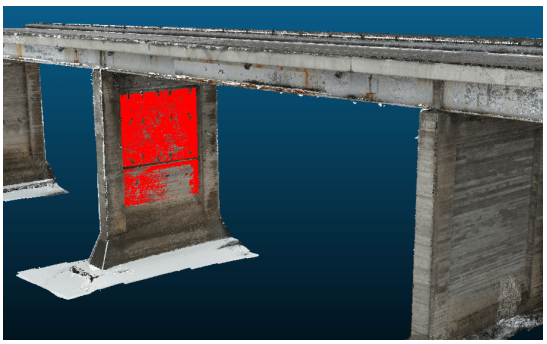
(a) Bjelke nedstrøms



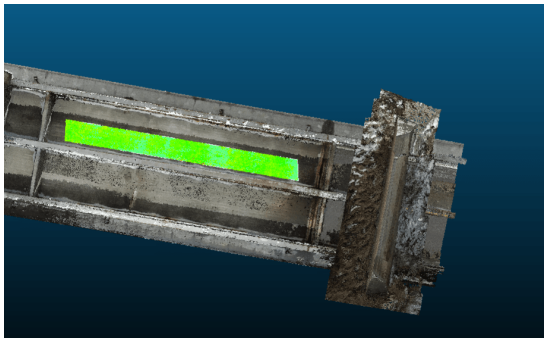
(b) Midtre bjelke



(c) Bjelke oppstrøms



(d) Pilar



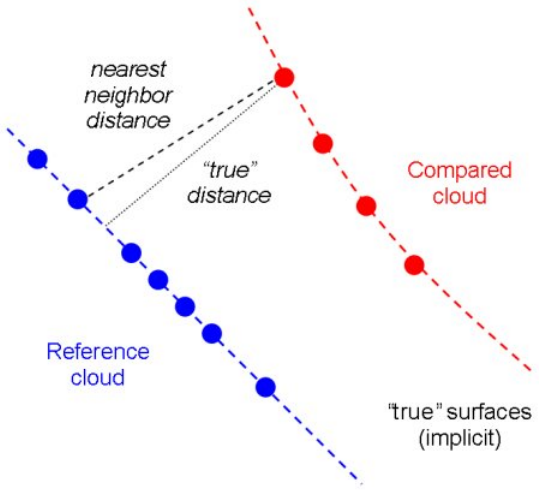
(e) Underside brudekke

Figur 3.7: De fem testflatene som ble valgt

Punktskyen fra det bakkebaserte laserscannet ga god dekning på disse områdene, derfor ble disse områdene valgt som testflater.

3.8.1 Metrisk evaluering av modell

For å evaluere modellen laget av fotogrammetri ble sky-til-sky-distanse brukt. Denne metoden måler avstander mellom punktskyer, og gir i dette tilfellet mål på hvor mye flatene fra figur 3.7 avviker fra referanseflaten (laserscan).



Figur 3.8: Distanse mellom to punktskyer

Punktskyen dannet av laserscan fikk i dette tilfellet en god del lavere punktetthet enn den fotogrammetriske punktskyen. Dette gjør at en nærmeste-nabo-distanse vil være unøyaktig (slik som vist i figur 3.8.1). CloudCompare sitt verktøy "Cloud-to-Cloud Distance" tillater en lokal modellering mellom punkter i referanseskyen, og vil måle avstanden til nærmeste punkt i denne modellen. CloudCompare har muligheten for mange typer lokal modellering. Modellen valgt her var et plan dannet av minste kvadraters metode basert på de seks nærmeste punktene.

3.8.2 Fremgangsmåte

Flatene ble klippet ut manuelt i CloudCompare. Etter det ble RANSAC-segmentering implementert. Dette ble gjort av to grunner:
-Hente ut riktig flate der det har oppstått duplikat-flater
-Fjerne støy fra flater

Segmenteringen ble gjort i Python.

3.9 Deteksjon av rust

For å detektere rust ble RGB-verdier fra modellene brukt. Dette ble gjort ved å manuelt lese RGB-verdier fra rust-punkter i modellene og bruke disse som "treningsdata". Ved å definere øvre og nedre terskel for rød, grønn og blå kan punkter eller områder filtreres ut innenfor disse terskelverdiene. Parallelt ble det forsøkt en klassifisering med Random Forest. Dette ble gjort i både punktsky og overflatemodell. Fordelen med å gjøre denne operasjonen i en overflatemodell er at det kan bli gjort nøyaktige areal-målinger av rust.

H-bjelkene ble manuelt klippet ut fra resten av brua i CloudCompare, dette var for å kunne gjøre rustdeteksjon utelukkende på stålflater.

3.9.1 Fargekriterier ved terskelverdier

For deteksjon med terskelverdier ble det samlet inn to sett med rustpunkter. Disse settene ble deretter brukt til å lage to sett med terskelverdier for å skille rustfargene fra andre farger. Disse terskelverdiene ble funnet ved å se på de øverste og nederste verdiene til rustpunktene. I tillegg ble det laget forholdstall mellom fargebåndene. For disse forholdstallene ble bare minsteverdiene brukt. Fargedataen for rustpunktene ligger i vedlegg A.

Ved innsamling ble det forsøkt å finne god variasjon i punktene. Det første settet med punkter ble samlet inn fra både rustavrenning og rustpunkter. Dette var for å finne en allsidig mengde punkter som skulle inngå i all synlig rust og rustavrenning. Ut i fra disse punktene ble den laveste og høyeste verdien for de ulike båndene samlet inn sammen med de laveste forholdstallene. Disse ble valgt som kriterier, og er oppsummert i tabell 3.5. Disse terskelverdiene blir heretter referert til som de milde terskelverdiene.

Tabell 3.5: Oversikt over de endelige kriteriene som ble brukt for å skille ut punkter med rust i den milde klassifiseringen.

Kriterie	**Nedre grense**	**Øvre grense**
rød	70	200
grønn	30	185
blå	0	140
rød/grønn	1.09	–
rød/blå	1.4	–
grønn/blå	1.15	–

Det andre settet med punkter ble samlet inn for å kunne segmentere rustområder som ikke inneholdt rustavrenning. Innsamlingen ble gjort på samme måte som i datasett en, men denne gangen ble det bare valgt punkter hvor det var faktisk korrosjon. I denne klassifiseringen ble terskelverdien kun basert på nedre grense i forholdstallene. Terskelverdiene som ble valgt ut i fra dette datasettet er vist i

tabell 3.6. Videre i denne oppgaven blir disse terskelverdiene omtalt som de strenge terskelverdiene.

Tabell 3.6: Oversikt over de strenge kriteriene som ble brukt for å skille ut korrosjonsområder.

Kriterie	**Nedre grense**
rød/grønn	1.45
rød/blå	1.85
grønn/blå	1.15

3.9.2 Filtrering av punktsky

For å identifisere de ulike rustpunktene ble det laget en Python-kode. I koden blir hvert punkt fra punktskyen importert, punktene består av koordinater (xyz) og RGB-verdier. Koden itererer gjennom alle punktene og filtrerer ut punkter basert på terskelverdiene som er gitt.

Python-koden ligger i vedlegg B.2.

3.9.3 Filtrering av overflatemodell

Overflatemodeller (mesh) er bygget opp av trekanter der hjørnepunktene består av koordinater (xyz) og fargeverdier (RGB). Koden fungerer på samme måte som med punktskyen, men denne koden filtrerer ut trekanter der alle hjørnepunktene (på grunnlag av RGB-verdier) har blitt klassifisert som rust.

Koden lager deretter en overflatemodell som kun består av trekanter klassifisert som rust. Da denne overflatemodellen er en tredimensjonal flate, er det mulig å regne arealet av den.

Python-koden som ble brukt ligger i vedlegg B.3.

3.9.4 Rustdeteksjon med Random Forest

Random Forest ble implementert for å undersøke om en maskinlæringsalgoritme ville gi en mer nøyaktig klassifisering av rust. Det ble laget Python-kode for dette.

Ved bruk av denne maskinlæringsalgoritmen var det behov for en større mengde treningsdata enn i de to andre metodene. Derfor ble en større mengde punkter med mørke rustfarger klippet ut fra punktskyen i CloudCompare. Det var fokus på at disse punktene ikke skulle være rustavrenning, men faktisk rust. Parallelt ble det også plukket ut punkter som ikke var rust. Det ble totalt klippet ut 176513 punkter som treningsdata. Deretter ble det kjørt en predikasjon på både punktskyen og overflatemodellen, med samme fremgangsmåte som beskrevet i 3.9.2 og 3.9.3.

Python-koden som ble brukt ligger i vedlegg B.4.

Kapittel 4

Resultater

4.1 Bildekvalitet

Figur 4.1 viser noen utvalgte bilder fra droneflygingen. Det blir gode detaljer/kontraster på alle bildene. Snø på bakken gir hvit lysrefleksjon, dermed lyser det godt opp under brua.

(a) Bilde ovenifra. Tatt med P1

(b) Skråbilde av brudekke. Tatt med P1

(c) Bilde nedenfra. Tatt med H20

(d) Bilde skrått under. Tatt med H20

Figur 4.1: Eksempler på bilder av Uvesund bru tatt under droneflyging

4.1.1 Blur i skråbilder

Figur 4.2 viser hvordan hjørnet av bildet er mer utsatt for blur. I dette tilfellet blir en rustflekk brukt som eksempelobjekt for å illustrere problemet. Problemet oppstår som følge av at den ene siden av brua er nærmere kameraet i bildesituasjonen, og siden kameraet er i bevegelse gir dette større sannsynlighet for blur.

(a) Skråbilde med rustflekk i midten av bildet

(b) Skråbilde med rustflekk i hjørne av bildet

(c) Rustflekk i midten av bilde

(d) Rustflekk i hjørne av bilde

Figur 4.2: Rustflekk vist fra to forskjellige skråbilder fra samme bildeserie. Det oppstår blur i bilde (d)

4.2 Resultat modellprosessering

4.2.1 Modeller

Det ble produsert totalt fire modeller fra forskjellige kombinasjoner av datasettet.

1. Modell med alle bildene, uten GCP-er. Her ble det brukt totalt 985 bilder, som resulterte i 496,740,678 punkter. Heretter referert til som **Modell 1**.
2. Modell med alle bildene, inkludert GCP-er. Her ble det brukt 985 bilder, som resulterte i 488,566,471 punkter. Heretter referert til som **Modell 2**.
3. Modell uten nadir-bilder, inkludert GCP-er. Her ble det brukt totalt 922 bilder, som resulterte i 350,031,037 punkter. Heretter referert til som **Modell 3**.
4. Modell uten bildene tatt rett under brua, inkludert GCP-er. Her ble det brukt totalt 845 bilder, som resulterte i 132,420,230 punkter. Heretter referert til som **Modell 4**.

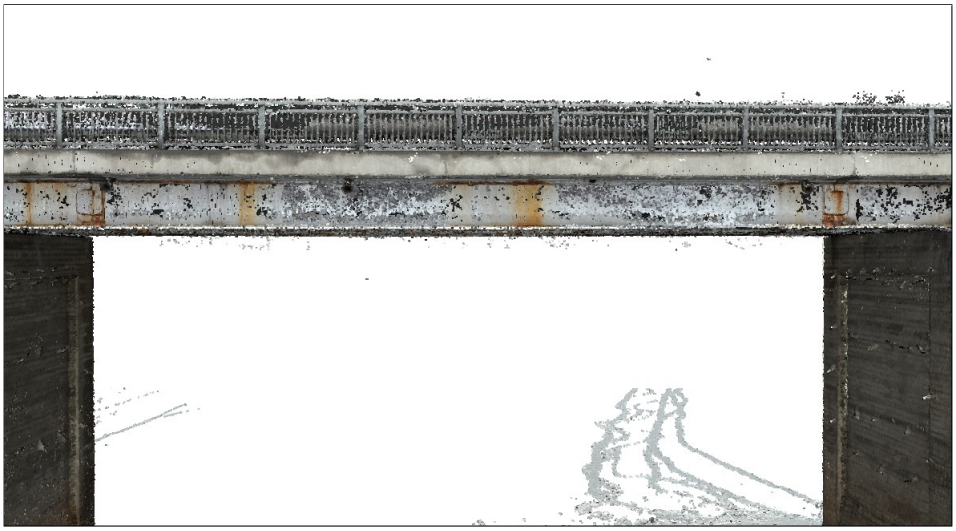
Figur 4.3 viser en overflatemodell av Modell 2.

Figur 4.3: Overflatemodell av Uvesund bru

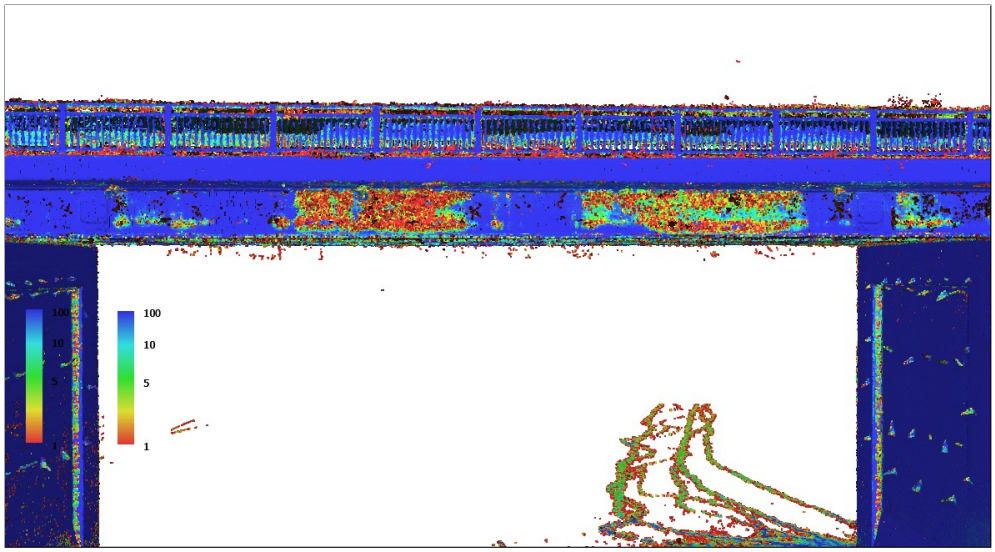
Figurene 4.4, 4.5, 4.6 og 4.7 viser et utsnitt av Uvesund bru vist fra samme posisjon i forskjellige digitale framstillinger.

Figur 4.4: Bilde tatt med P1-kameraet

Figur 4.5: Overflatemodell laget av Modell 2



Figur 4.6: Punktsky (Modell 2)



Figur 4.7: Punktsky farget etter konfidensnivå. Mørkeblå er høy konfidensverdi og rød er lav.

Figur 4.5 og 4.6 viser hvordan flatene har blitt rekonstruert i modeller. Overflatemodellen har hull i stålbjelkene, men har tilsynelatende en fornuftig geometri. Punktskyen bærer preg av støy og hull.

Hullene i nevnte modeller befinner seg på H-bjelkene. Ved å sammenligne bildet vist i 4.4 med punktskyen i 4.6, sammenfaller områdene med hull i modellen og homogene flater. Med homogene flater menes det flater med ensartet farge og lite tekstur/mønster. Med homogene flater i bildene vil bildematching-algoritmen finne færre sammenbindingspunkter i disse områdene. 3D-modellen vil da bli mindre tett i disse områdene. Konfidensnivået belyser hvordan de mest homogene områdene har lavest konfidensnivå.

4.2.2 Prosesseringstid

Tabell 4.1: Prosesstider for de forskjellige modellene.

Prosess	Modell 1	Modell 2	Modell 3	Modell 4
Alignment	110 min	91 min	79 min	65 min
Depth Maps	218 min	216 min	259 min	202 min
Point Cloud	434 min	463 min	408 min	447 min
Totalt	**762 min**	**770 min**	**746 min**	**714 min**

Tabell 4.2: Spesifikasjoner for PC brukt til prosessering

	Spesifikasjon
Prosessor	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
RAM	32,0 GB
Skjermkort	NVIDIA Quadro T1000
Operativsystem	Windows 10

4.2.3 RMSE verdier

Prosessen i Agisoft produserte RMSE-verdier basert på forskjellen mellom GCP-er og modell etter fullstendig produksjon av punktskyer. Tabell 4.3 og 4.4 viser RMSE-verdier for henholdsvis Modell 2 og Modell 3.

Tabell 4.3: RMSE-verdiene for modellen med alle bildene (Modell 2)

Punkt	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Bilde (piksel)
GCP1	-6.55857	-1.26271	-0.917412	6.74173	1.016
GCP2	0.1729	1.82001	-0.766921	1.98255	1.541
GCP3	-0.832326	2.86288	-5.08595	5.8954	2.559
GCP5	-6.38193	-0.741835	-0.694809	6.46236	0.659
Total	**4.59528**	**1.84752**	**2.63531**	**5.61024**	**1.633**

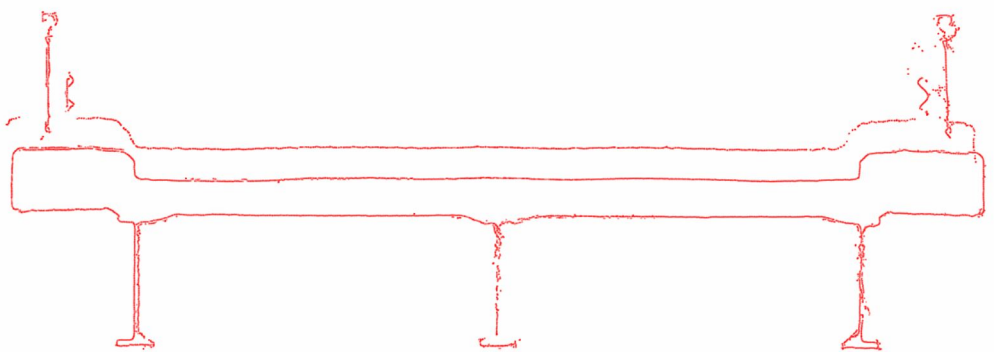
Tabell 4.4: RMSE-verdiene for modellen uten nadir-bildene (Modell 3)

Punkt	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Bilde (piksler)
GCP2	0.158303	-1.22755	-1.75453	2.14716	1.905
GCP3	-0.588011	1.16424	-3.88937	4.10225	0.863
Total	**0.430591**	**1.19632**	**3.01709**	**3.27405**	**1.734**

I forkant ble det totalt satt ut fem GCP-er. GCP nr. 4 ble ikke synlig nok på bildene grunnet slaps i veibanen. I Modell 3 ble kun to GCP-er synlige på bildene.

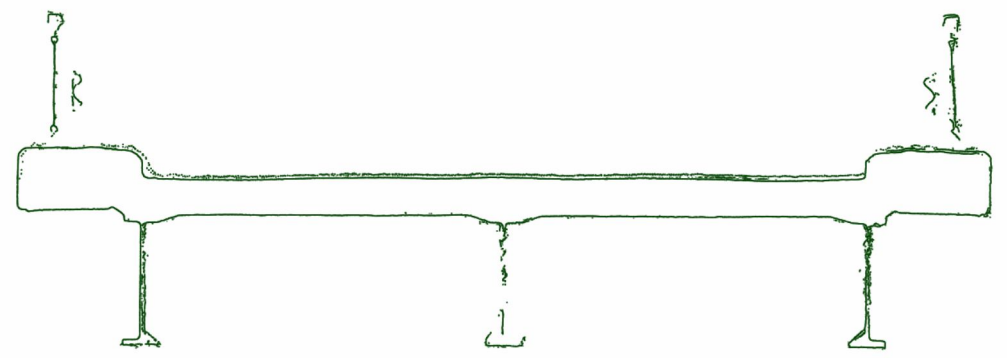
4.2.4 Tverrsnitt punktskyer

I dette delkapittelet presenteres tverrsnitt fra samme sted på de respektive modellene.



Figur 4.8: Tverrsnitt fra Modell 1

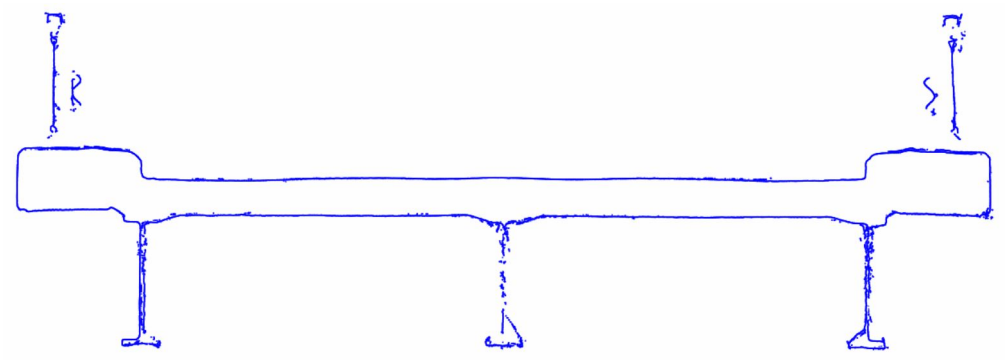
I figur 4.8 vises tverrsnittet fra Modell 1 (alle bilder uten GCP-er). Fra figuren er det tydelig at den øverste delen av brudekket har oppstått som duplikat-flate. Her har ikke skråbildene på brudekket sammenfalt med nadir-bildene i modellen. H-bjelken i midten består tydelig kun av én flate, i motsetning til bjelken til venstre som har fått fullstendig geometri.



Figur 4.9: Tverrsnitt fra Modell 2

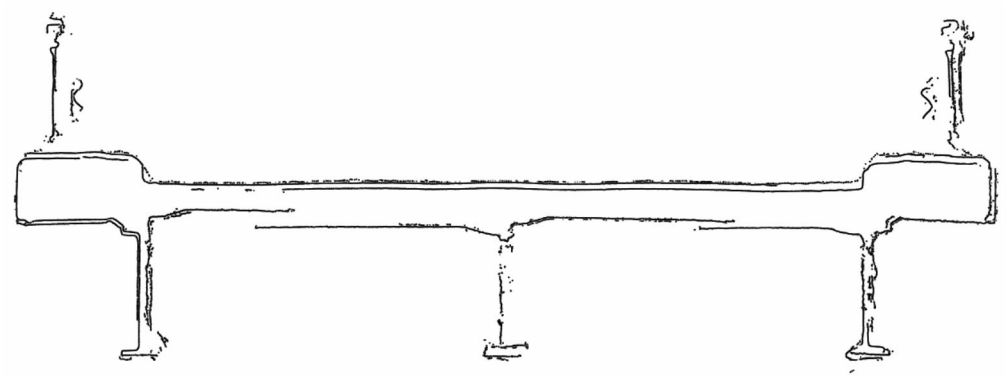
I figur 4.9 vises et tverrsnitt av Modell 2 (alle bilder med GCP-er). Her har modellen fortsatt en duplikat-flate øverst, men disse sammenfaller i større grad enn Modell 1.

H-bjelkene er er forholdsvis like i Modell 1 og Modell 2.



Figur 4.10: Tverrsnitt fra Modell 3

I figur 4.10 vises et tverrsnitt av Modell 3 (nadir-bildene eksludert). I denne modellen er det ingen duplikat-flate, og brudekket har en fornuftig geometri.

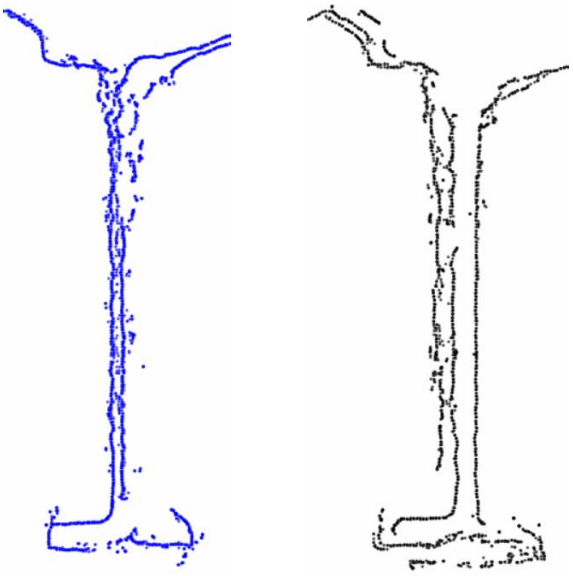


Figur 4.11: Tverrsnitt fra Modell 4

I figur 4.11 vises tverrsnittet fra Modell 4 (bildene fra rett under brua er ikke brukt). I toppen av brudekket er ikke flatene kontinuerlige. H-bjelkene har betydelig dårligere geometri sammenlignet med de tidligere modellene.

4.2.5 Resultater av H-bjelker

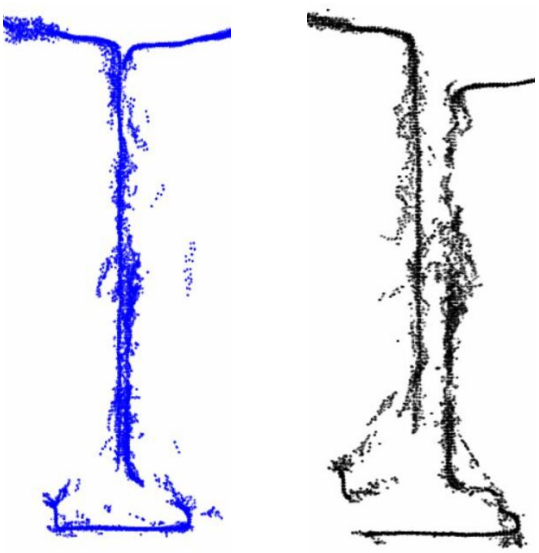
I figur 4.12 og 4.13 vises tverrsnitt av H-bjelkene i Modell 3 og 4 på to forskjellige steder på brua. I begge sammenligningene er det tydelig at geometrien i Modell 3 sin H-bjelke er den mest riktige av de to.



(a) Ytre H-bjelke i Modell 3.

(b) Ytre H-bjelke i Modell 4.

Figur 4.12: Tverrsnitt av ytre H-bjelke på Modell 3 og 4. På Modell 4 er det tydelig at det oppstår duplikat-flater i bjelken. Den har også en forskyvning i geometrien som fører til at bjelken ikke sammenfaller på begge sidene.



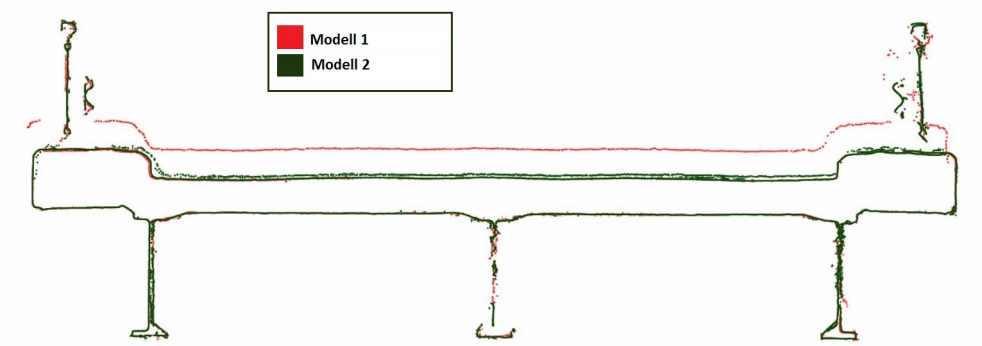
(a) Midtre H-bjelke ved Modell 3.

(b) Midtre H-bjelke ved Modell 4.

Figur 4.13: Tverrsnitt av midtre H-bjelke ved Modell 3 og 4. Her er det mye støy i begge punktskyer. Samtidig blir bjelken i Modell 4 mye breiere enn den skal være. Forskyvningen nevnt tidligere blir enda mer synlig her.

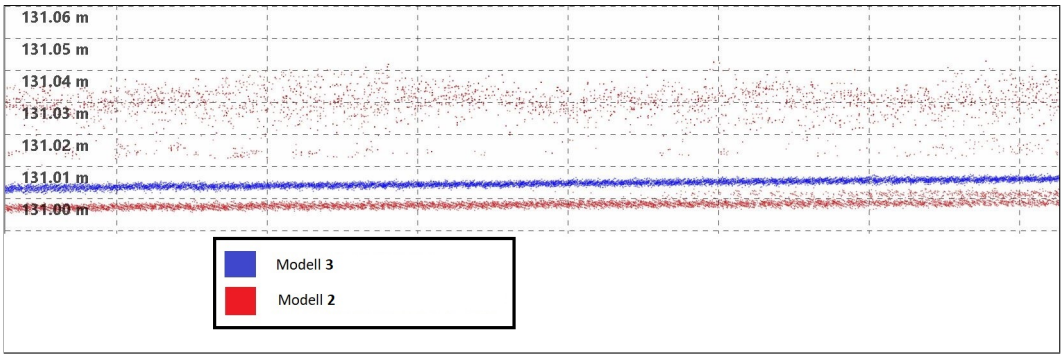
4.2.6 Sammenligning av tverrsnitt

I figur 4.14 er tverrsnittene av Modell 1 og Modell 2 presentert i samme figur. Duplikat-flatene er synlig i begge modellene, men Modell 1 sin øverste flate er betraktelig mer feilplassert da den ligger for høyt. Fra Modell 1 til Modell 2 er det tydelig at GCP-ene har bidratt til å dytte duplikat-flaten ned til riktig høyde.



Figur 4.14: Tverrsnitt av Modell 1 og Modell 2

I figur 4.15 gis det et nærbilde av tverrsnittene over asfalten i Modell 2 og Modell 3. Her er den den øverste flaten i Modell 2 (duplikat-flate) betraktelig mer støyete enn flaten i Modell 3.



Figur 4.15: Sammenligning av veibaneoverflaten i Modell 2 og 3.

4.2.7 Kjennetegn ved de forskjellige modellene

Tabell 4.5 viser en oversikt over tydelige kjennetegn ved de forskjellige modellene. Modell 3 har en tydelig sammenhengende modell, tynne bjelker og ingen dobbel asfaltlinje. Det kan derfor sies at modell 3 har flest av de ønskede karakteristikkene som samenfaller best med den virkelige brua.

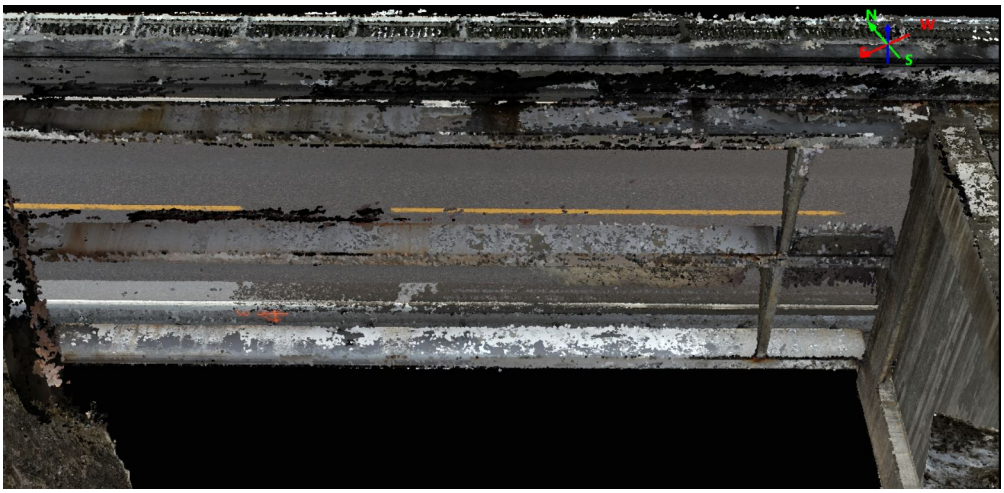
Tabell 4.5: Tydelige kjennetegn ved de forskjellige modellene

Modell:	**Sammenhengende modell**	**Dobbel asfaltlinje**	**Tynne bjelker**
Modell 1	Ja	Ja	Ja
Modell 2	Ja	Ja	Ja
Modell 3	Ja	Nei	Ja
Modell 4	Nei	Ja	Nei

4.2.8 Mangler i modellene

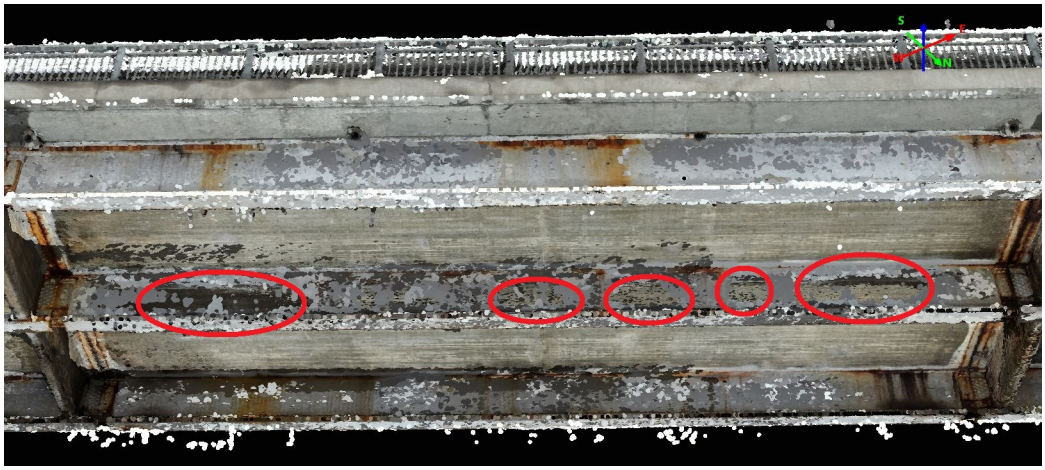
I figur 4.17 vises enden av brua i Modell 2. I disse partiene har geometrien blitt ufullstendig i alle modellene. På begge endene ble det færre bildevinkler grunnet naturlige hindringer. Dette førte til at disse partiene ble preget av støy og mangler.

Figur 4.16: Område hvor modellen har blitt mangelfull. Dette gjelder også på andre siden av brua.



Figur 4.17: Modellen i enden av brua. "Himlingen" under brua mangler, derfor sees det rett opp til asfalten. Bjelkene framstår som ufullstendige og støyete.

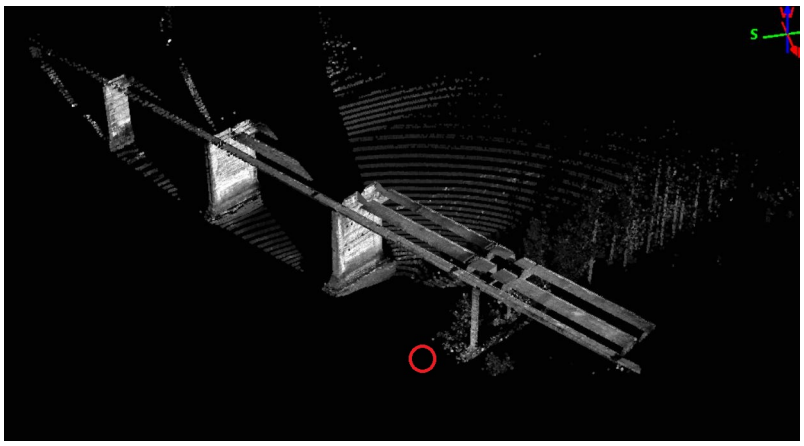
I figur 4.18 vises et parti fra midten av brua der det er mye hull i punktskyen. De fleste hullene forekommer på H-bjelkene.



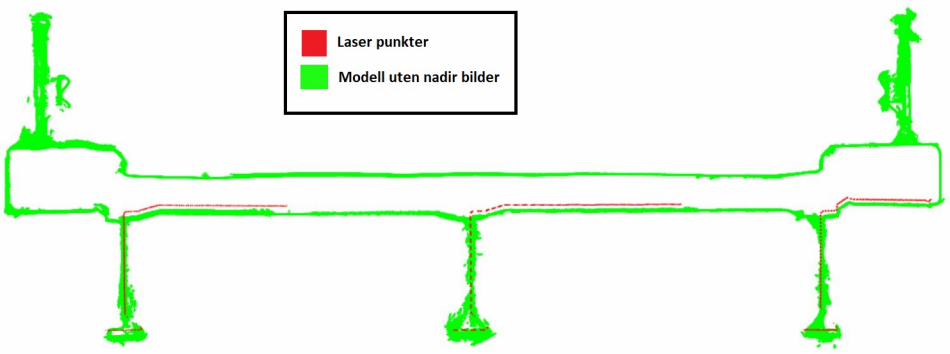
Figur 4.18: Hull i punktskyen på den midterste H-bjelken. Hullene er markert med røde sirkler.

4.3 Laserscan

Punktskyen dannet av laserscanning besto av totalt 123 326 punkter. I figur 4.19 er det synlig at kun deler av brua blir med i denne punktskyen. Bjelkene og pilarene ble godt dekket, og hadde tilstrekkelig punkttetthet til videre analyse.



Figur 4.19: Scan fra totalstasjon på bakken. Rød ring markerer oppstillingen.

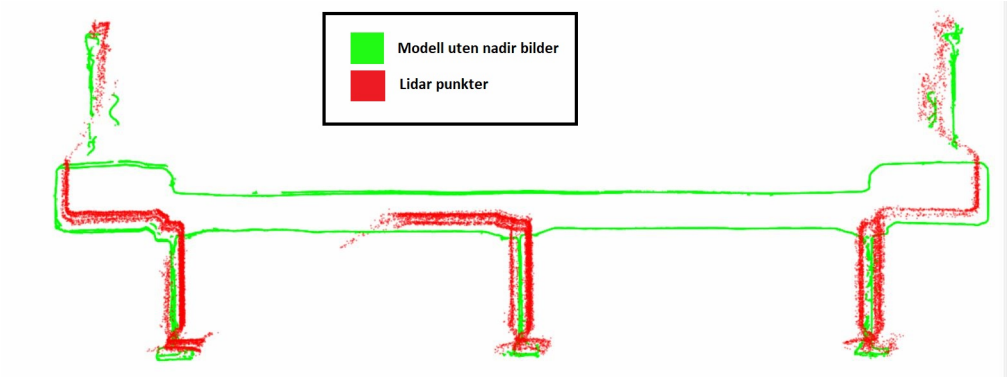


Figur 4.20: Tverrsnitt av brua. Fotogrammetrimodell sammenlignet med punktsky dannet av bakkebasert laserscan.

Figur 4.20 gir et grovt overblikk over hvordan de to modellene sammenfaller. I grunnriss sammenfaller modellene bra, men i høyde sees avvik under brua. En grundigere analyse beskrives i avsnitt 4.5.

4.4 LiDAR fra drone

I figur 4.21 vises den resulterende punktskyen fra drone-LiDAR sammen med Modell 3. LiDAR-scannet fremstår her som betydelig upresis og preget av støy.



Figur 4.21: Modell fra fotogrammetri sammenlignet med modell dannet av drone-LiDAR.

LiDAR-punktskyen klarer ikke å angi flater, og er uegnet til videre sky-til-sky-analyse grunnet dårlig kvalitet.

4.5 Sammenligning laserscan og fotogrammetri

Flatene som er brukt i denne sky-til-sky-analysen er beskrevet i kapittel 3.8.

4.5.1 Punkttetthet

Tabell 4.6 viser punkttettheten over områdene som ble brukt i analysen.

Tabell 4.6: Punkttettheter på modellen dannet av laserscan sammenlignet med fotogrammetri-modellen.

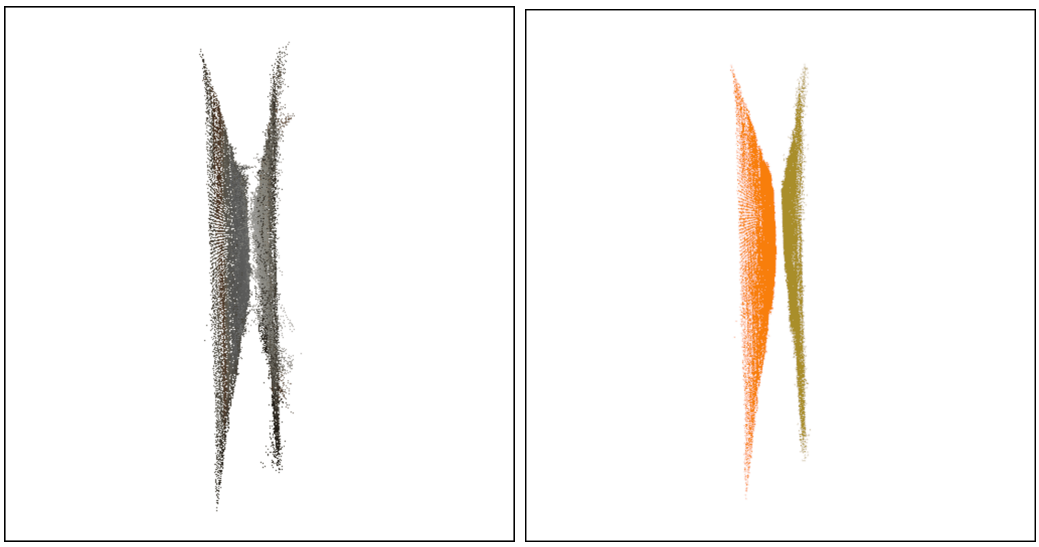
Flate	**Laserscan (p/m^2)**	**Fotogrammetri-modell (p/m^2)**
Bjelke nedstrøms	641	13589
Midtre bjelke	493	5878
Bjelke oppstrøms	300	6699
Pilar	231	12338
Under brudekke	337	6167

4.5.2 RANSAC i pre-prosessering

Som pre-prosessering til videre analyse ble RANSAC implementert for å isolere flater i punktskyene. Dette ble gjort for å gjøre det mulig å sammenligne den endelige mo-

dellen med punktskyen generert av laserscan. Mange av flatene inneholdt støy som gjorde det vanskelig å bruke flatene direkte i sky-til-sky-analyse. Eksempelvis hadde mange av stålbjelkene støy som gjorde det vanskelig å skille sidene fra hverandre. På pilarene under brua stakk det ut armerings-stål som ville skapt feil resultater i sammenligning med laserscan.

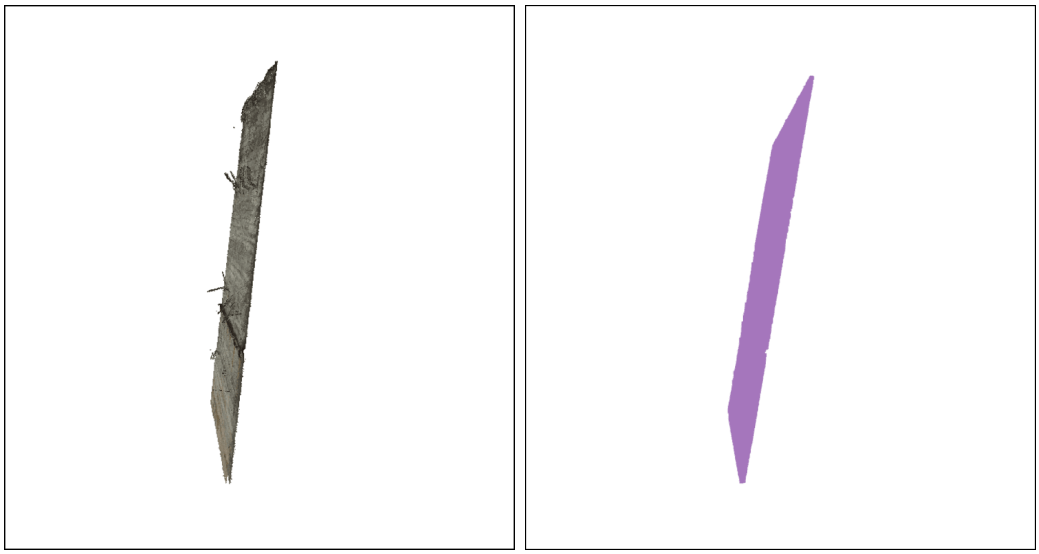
Figur 4.22 og 4.23 er to eksempler på flater der RANSAC ble implementert:



(a) Utsnitt av bjelke. Flatene her inneholder mye støy.

(b) Samme flater etter at RANSAC er implementert.

Figur 4.22: RANSAC implementert på bjelke. Parametre: *Minimumsforhold* = 0.5, *Terskelverdi* = 0.005



(a) Flate på pilar før RANSAC

(b) Samme flate etter RANSAC

Figur 4.23: RANSAC implementert på flate fra pilar. Parametre: *Minimum ratio* = 0.8, *Threshold* = 0,02

4.5.3 Sky-til-sky-distanse

I denne analysen ble sky-til-sky-distanse brukt for å måle nøyaktigheten til tre forskjellige punktskyer. Alle skyene ble målt opp mot punktskyen fra lasercannet. I tabellene nedenfor betyr variabelen "Gj.snitt" gjennomsnittlig avstand mellom referansesky og modellen. RMSE er det statistiske begrepet Root Mean Square Error. Den totale RMSE-verdien er regnet ut ved å ta gjennomsnittet av RMSE-verdiene.

Tabell 4.7: Sky-til-sky-distanser med Modell 4

Flate	**Gj.snitt (m)**	**RMSE (m)**
Bjelke nedstrøms	0.025	0.026
Midtre bjelke	0.023	0.024
Bjelke oppstrøms	0.021	0.021
Pilar	0.008	0.008
Under brudekke	0.018	0.018
Total		**0.019**

Tabell 4.8: Sky-til-sky-distanser med Modell 2

Flate	**Gj.snitt (m)**	**RMSE (m)**
Bjelke nedstrøms	0.015	0.016
Midtre bjelke	0.012	0.014
Bjelke oppstrøms	0.008	0.009
Pilar	0.019	0.022
Under brudekke	0.053	0.052
Total		**0.023**

Tabell 4.9: Sky-til-sky-distanser med Modell 1

Flate	**Gj.snitt (m)**	**RMSE (m)**
Bjelke nedstrøms	0.019	0.019
Midtre bjelke	0.012	0.012
Bjelke oppstrøms	0.011	0.012
Pilar	0.017	0.026
Under brudekke	0.055	0.054
Total:		**0.025**

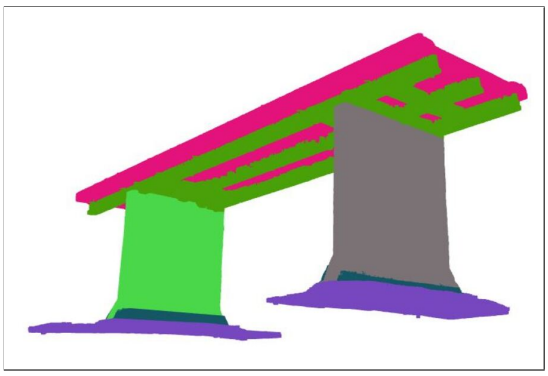
Modell 4 gir den laveste totale RMSE-verdien. Dette er modellen som er sammenstilt uten bildene fra undersiden av brua. Den får lavere verdier på pilaren og under brudekke, men på bjelkene får den høyere RMSE-verdier enn de andre modellene. Modell 2 og Modell 1 får veldig like verdier. I disse to modellene er de samme bildene brukt, men i Modell 1 er det ikke brukt GCP-er.

Det er viktig å notere at alle sammenligningene gjort i dette delkapittelet ikke er på nøyaktig samme arealer for hver modell. Dette vil defineres som en liten systematisk feil i denne metoden og vil gi noe avvik i sammenligningene. RMSE-verdiene kan fortsatt sammenlignes med ganske god pålitelighet.

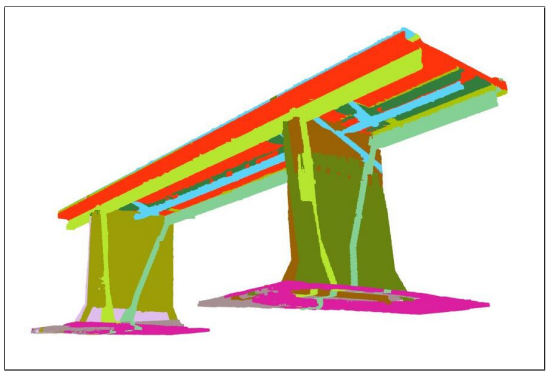
4.6 RANSAC-segmentering

Figur 4.25 viser resultatene fra forskjellige terskelverdier i RANSAC-segmentering. Formålet var å få til en automatisk segmentering av stålbjelkene i sin helhet.

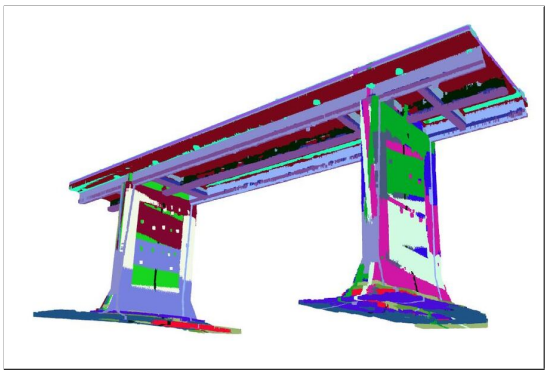
Figur 4.25a viser den terskelverdien som ga det nærmeste resultatet til en fullstendig isolasjon av stålbjelkene.



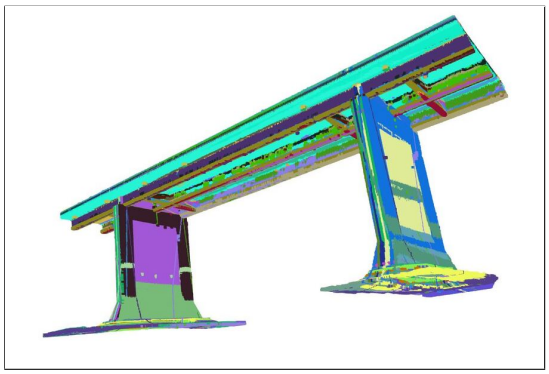
(a) Segmentering med terskelverdi 0.5



(b) Segmentering med terskelverdi 0.25

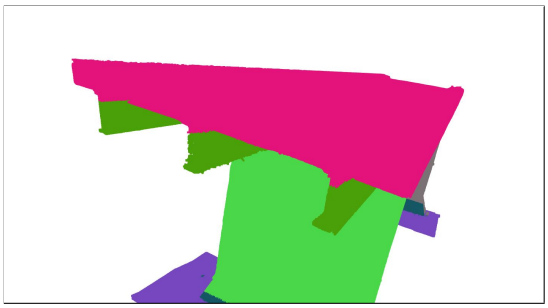


(c) Segmentering med terskelverdi 0.1

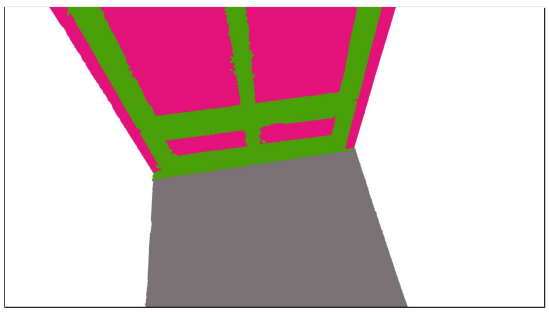


(d) Segmentering med terskelverdi 0.05

Figur 4.24: Flatesegmentering med RANSAC ved forskjellige terskelverdier. Parameteren "minimumsforhold" ble holdt konstant på 0.01.



(a) De øverste delene av H-bjelkene ligger i samme segment som brudekke.



(b) Øverste del av pilarene ligger i samme segment som H-bjelkene.

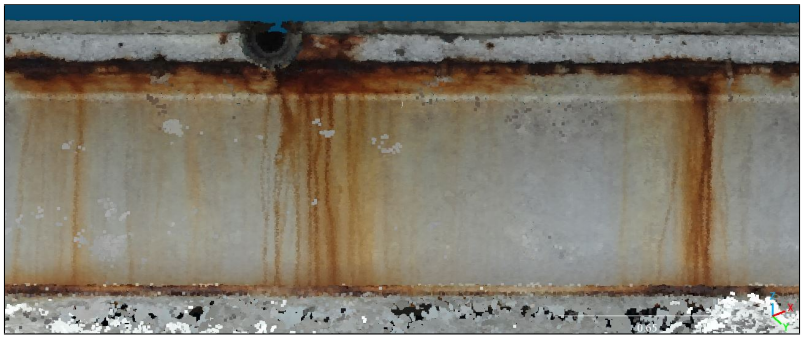
Figur 4.25: Eksempler på at terskelverdi 0.5 ikke fullstendig segmenterer H-bjelkene.

4.7 Deteksjon av rust

Det ble totalt gjort tre deteksjoner av rust basert på RGB-verdier. Det ble testet to ulike sett med terskelverdier: et sett med strenge kriterier og et med milde kriterier. Samtidig ble det gjort en klassifisering med Random Forest.

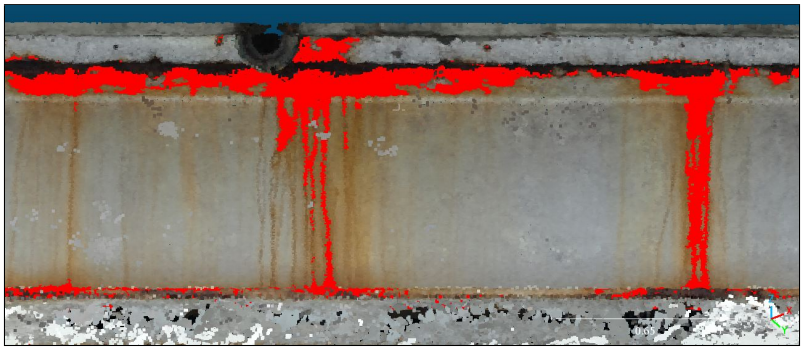
4.7.1 Punktsky

Figurene 4.26 og 4.27 viser områdene hvor rust har blitt klassifisert på H-bjelkene. Figurene viser både bruk av de milde og strenge terskelverdiene, samt resultatet av klassifisering med Random Forest. Alle klassifiseringene av rust er illustrert i rødt.



(a) H-bjelke i ytterkant av brua med tydelig rust i toppen, samt avrenning av rust ned på bjelkene.

(b) Klassifisert rust ved milde terskelverdier.

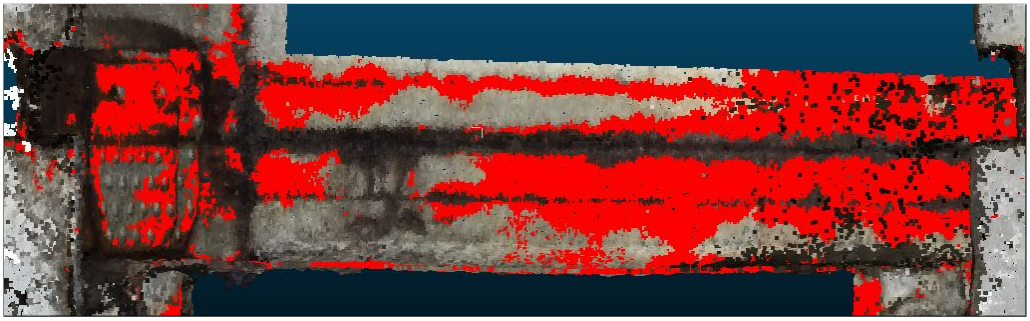


(c) Klassifisert rust ved strenge terskelverdier

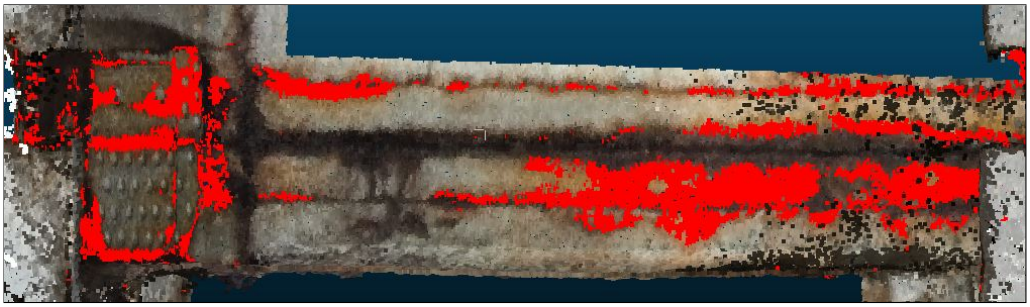
(d) Klassifisert rust med Random Forest

Figur 4.26

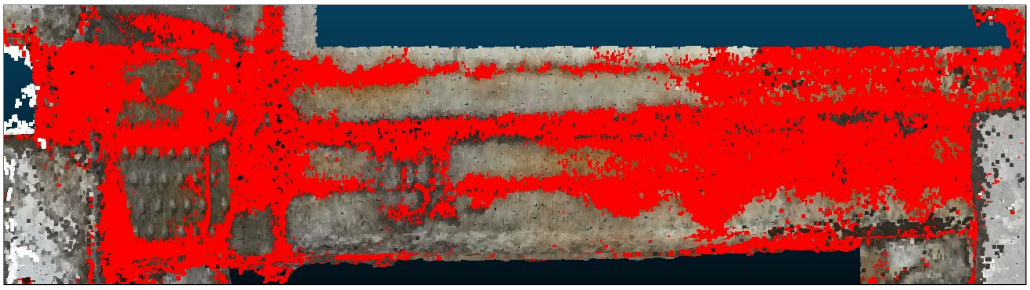
(a) Stålbjelken mellom H-bjelkene, sett fra undersiden.



(b) Klassifisert rust ved milde terskelverdier.



(c) Klassifisert rust ved strenge terskelverdier.



(d) Klassifisert rust med Random Forest.

Figur 4.27

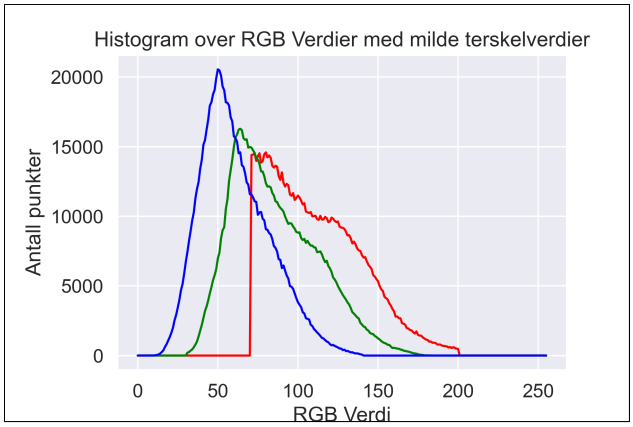
Tabell 4.10 viser mengden punkter som har blitt klassifisert i de forskjellige forsøkene. Det totale antallet punkter er antall punkter stålbjelkene utgjør i punktskyen.

Tabell 4.10: Antall punkter som er klassifisert som rust på H-bjelkene

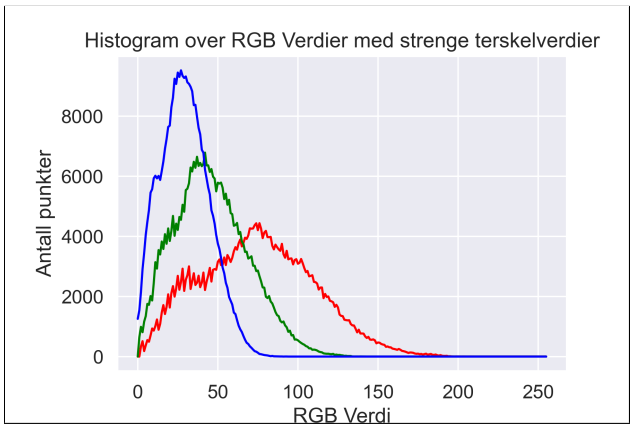
Terskelverdier	**Punkter detektert**	**Punkter totalt**
Mild	930 307	15 081 779
Streng	374 434	15 081 779
Random Forest	1 655 080	15 081 779

4.7.2 Fordeling av RGB-verdier i klassifisering

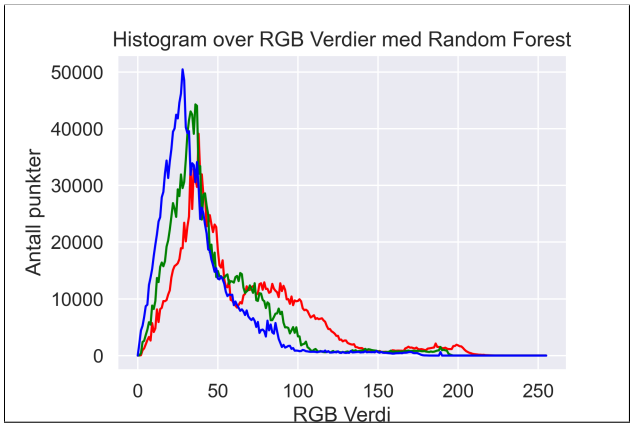
Figur 4.28 viser fordelingen av røde, grønne og blå verdier i punktene som er klassifisert som rust. Figurene 4.28a og 4.28b er fra deteksjonene med milde og strenge terskelverdier, og 4.28c er fra deteksjonen med Random Forest. Y-aksen viser den totale mengden punkter for hver fargeverdi.



(a) Mengden rød, grønn og blå i punktene klassifisert som rust med milde terskelverdier.



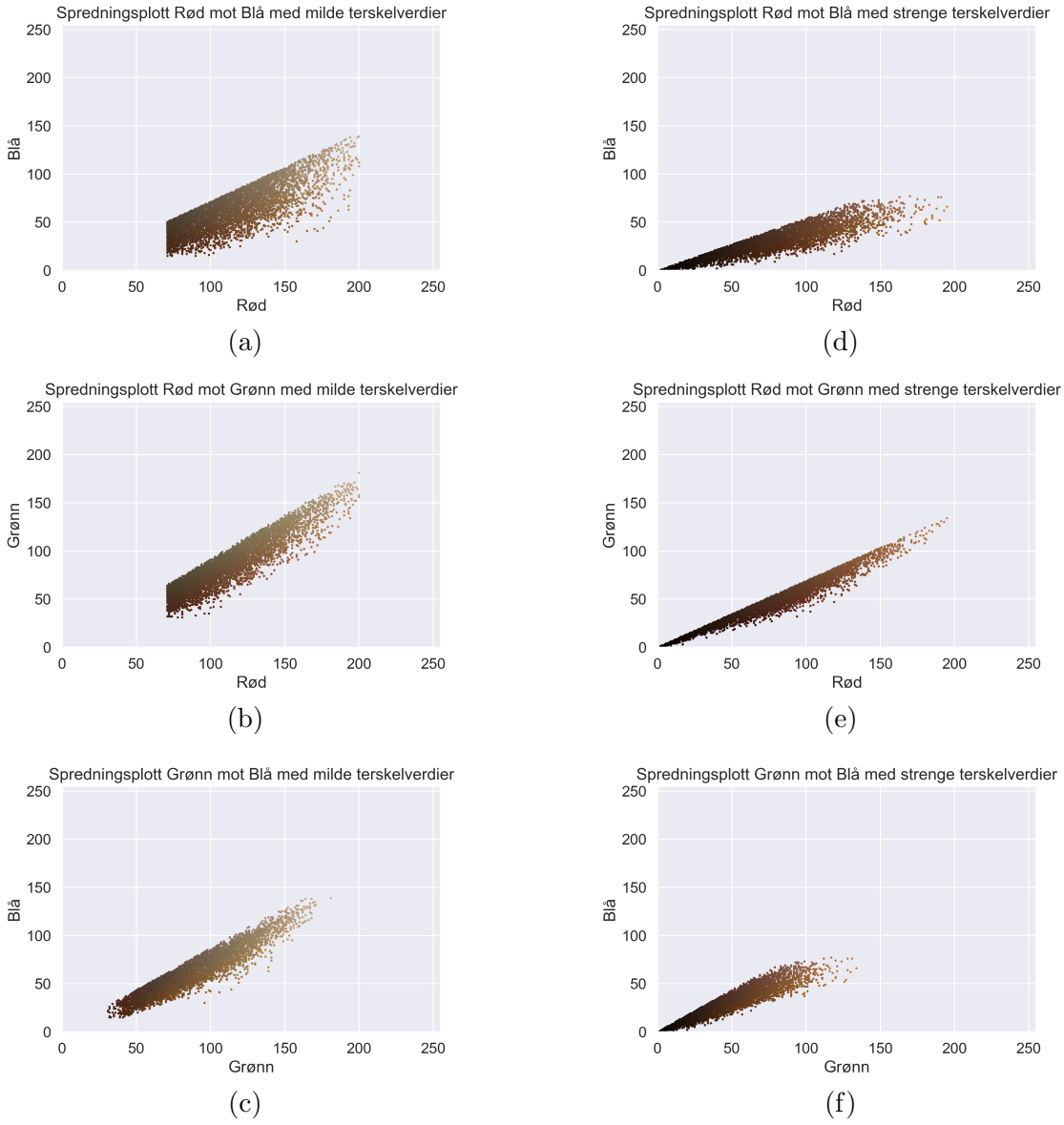
(b) Mengden rød, grønn og blå i punktene klassifisert som rust med strenge terskelverdier.



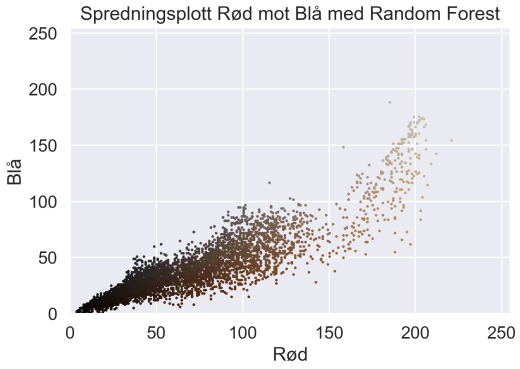
(c) Mengden rød, grønn og blå i punktene klassifisert som rust med Random Forest.

Figur 4.28

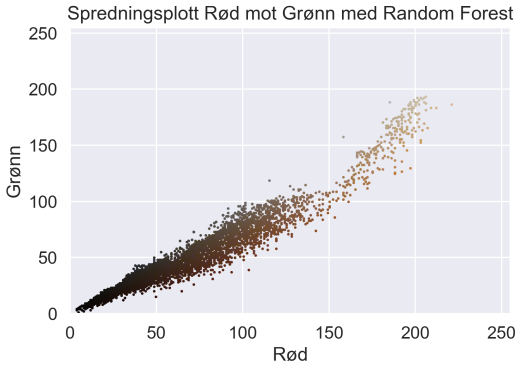
Figur 4.29 viser spredningsplott der farger i klassifiserte rustpunkter er plottet opp mot hverandre. Punktene i plottet er farget ut i fra sine sanne farger. Figurene for den milde klassifiseringen er i kolonnen til venstre, og den strenge klassifiseringen er i kolonnen til høyre. Figur 4.30 viser spredninsplottene fra Random Forest-klassifiseringen.



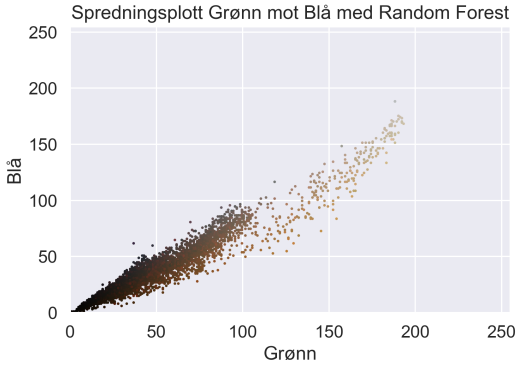
Figur 4.29



(a)



(b)

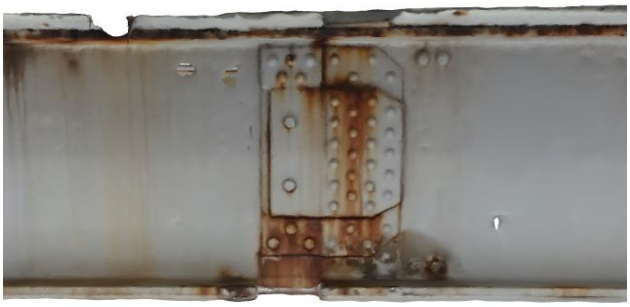


(c)

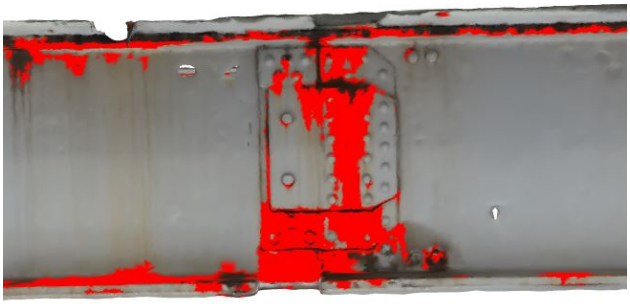
Figur 4.30

4.7.3 Rustdeteksjon i overflatemodell

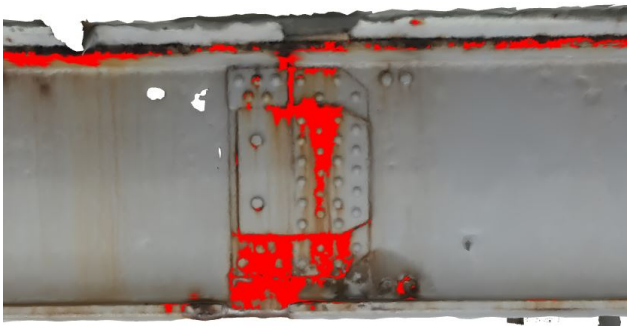
Figur 4.31 viser deteksjon av rust i overflatemodellen. I figurene 4.31b og 4.31c er det brukt milde og strenge terskelverdier, og i figur 4.31d er det brukt Random Forest.



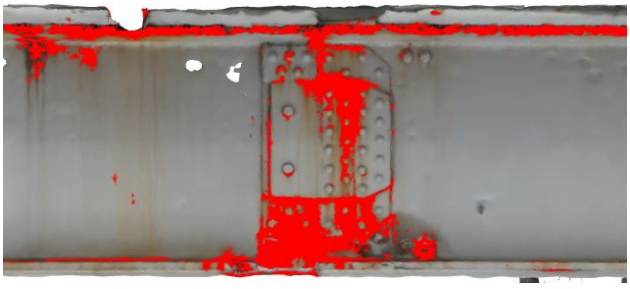
(a) Utsnitt av H-bjelke



(b) Milde terskelverdier



(c) Strenge terskelverdier



(d) Områder klassifisert av Random Forest

Figur 4.31

Tabell 4.11 viser resultatet av rustdeteksjon i overflatemodellen. Det teoretiske arealet er beregnet ut i fra dimensjonene til H-bjelkene, og er bare omtrentlig estimat på arealet. Dette arealet antas å være et bedre estimat på det totale arealet enn arealet gitt av overflatemodellen, da denne har hull og forvrengninger i visse partier.

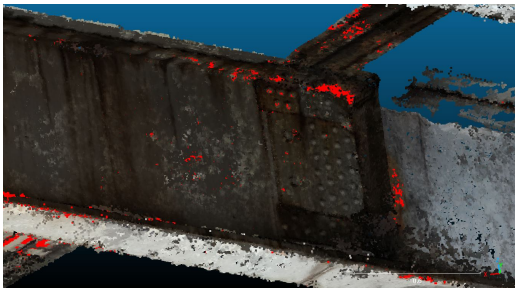
Tabell 4.11: Resultatene fra rustdeteksjon på overflatemodellen. Totalt areal er det teoretiske arealet til H-bjelkene.

Terskelverdier	**Detektert rust-areal (m^2)**	**Totalt areal (m^2)**	**Prosentandel**
Mild	27.85 m^2	843,9 m^2	3.30%
Streng	6.17 m^2	843,9 m^2	0.70%
Random Forest	45.49 m^2	843,9 m^2	5.39%

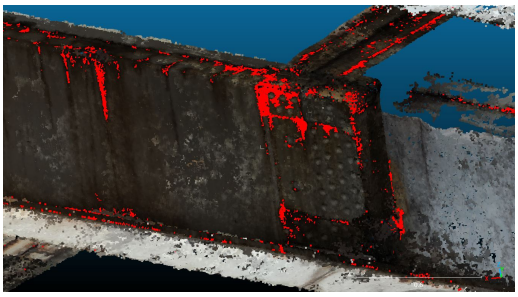
4.7.4 Feilklassifisering

Figur 4.32 viser at det er vanskelig å gjøre en nøyaktig klassifisering av rust i de mørkeste områdene på modellen. Den milde deteksjonen får nærmest ikke med seg noen rustpunkter. De strenge terskelverdiene klarer å detektere noe, men fremstår som ganske unøyaktig. Resultatet fra Random Forest viser en grov overklassifisering.

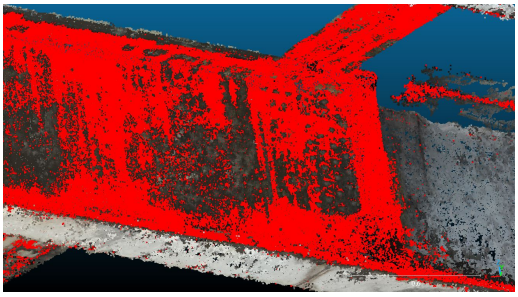
(a) Utsnitt av bilde brukt i fotogrammetriprosessen.



(b) Milde terskelverdier på modell



(c) Strenge terskelverdier på modell



(d) Områder klassifisert av Random Forest på modell

Figur 4.32: Klassifisering av rust under brua

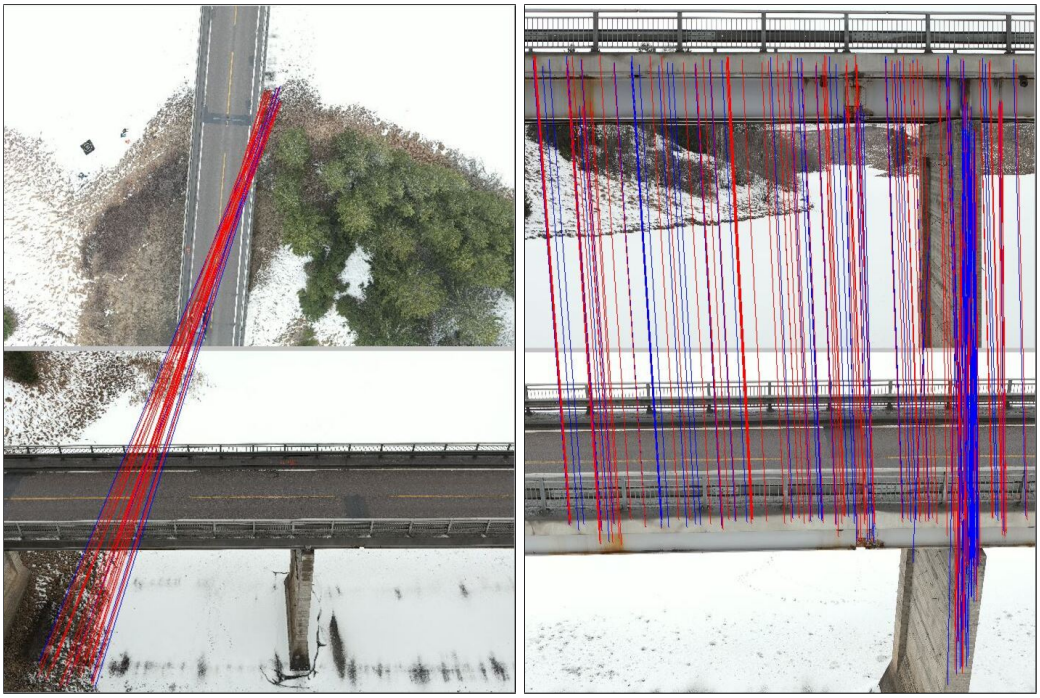
Kapittel 5

Diskusjon

5.1 Modell

5.1.1 Overordnet inntrykk av modellene

Resultatene fra tverrsnittene og oversikten i tabell 4.5 tilsier at Modell 3 ga den mest pålitelige representasjonen av brua. Dette er modellen som bruker alle bildene, med unntak av bildene tatt i nadir-posisjon. Det som skiller Modell 3 fra Modell 1 og 2 er duplikat-linjen over asfaltdekket som er vist i figur 4.8 og 4.9. Videre analyse viser at Agisoft sliter med å matche nadir-bildene med skråbildene av brudekket. Dermed oppstår det to separate flater. Det er ingen åpenbar forklaring på årsaken. En mulighet er at forskjellen i oppløsning blir så stor i de to bildene at algoritmen ikke fungerer optimalt. Som eksempel vil en vanlig bildematching basert på en SIFT-algoritme i prinsippet håndtere endringer i skala, men i dette tilfellet er endringen såpass stor at det kan ha negativ innvirkning. Samtidig er vinkelen mellom posisjonen der bildene er tatt stor og lysforholdene forskjellig i de to bildeseriene. I figur 5.1 vises det eksempel på at det ikke blir detektert fellespunkter i de to ulike bildevinklene. Den øverste flaten i Modell 1 (den som avviker mest i høyde) vil kun være modellert av nadir-bildene. Grunnen til at denne flaten avviker såpass i høyde kan være den svake geometrien nadir-bildene har.



(a) (b)

Figur 5.1: (a) Ingen fellespunkter på brua funnet mellom nadir-bildene og skråbildene, det er kun funnet fellespunkter på bakken. (b) Eksempel på mange fellespunkter funnet på brua i en annen bildeserie.

Generelt viser alle resultatene at H-bjelkene ikke har blitt fullstendig modellert. Den største årsaken til dette er de homogene flatene på H-bjelkene. Fotogrammetriprosessen er avhengig av å klassifisere mange fellespunkter i bildepar for å lage en tett punktsky. Mange partier på stålbjelkene på Uvesund bru er ensfargede flater med lite til null tekstur. Dette gjør at SfM-metoden får lite grunnlag for å lage modell av disse flatene.

Det er sannsynlig at en økning i antall bilder vil gi bedre resultater. Hvis det i tillegg hadde vært mulig å komme nærmere bjelkene ville det bli bedre romlig oppløsning på bildene av bjelkene, noe som ville gjøre det lettere å finne teksturer i flatene. Dette vil komplisere flygingen betraktelig, samtidig som det øker flytiden. Dermed er det kompromiss som må inngås, og det er ikke garanti for at resultatet blir signifikant bedre. Under brua er det også begrenset med mulige bildevinkler.

På en annen side er partiene med rustfarger godt modellert. I disse partiene er det rikelig med tekstur og kontraster, noe som gjør at algoritmene lettere finner fellespunkter i bildepar. Derfor gir dette godt grunnlag for arealberegninger på slike områder. På samme måte som rustflekker, gir også betong gode flater for modellering. Betongflatene i modellen fremstår som geometrisk komplette, og har få hull og lite støy.

Resultatene fra modellprosesseringen viser at det er viktig å inkludere bildene tatt fra rett under brua. Den visuelle inspeksjonen av Modell 4 påpeker den betydelige svakere geometrien som oppstår uten disse bildene.

Modellen ble dårlig modellert på undersiden i endene av brua, noe som drøftes videre i 5.2.1.

5.1.2 Georeferering

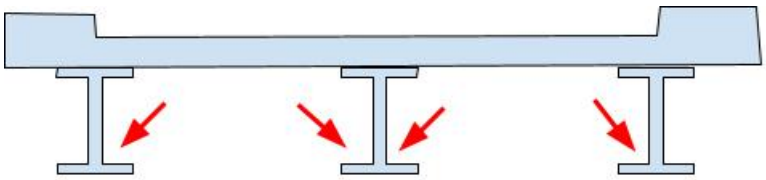
I tabell 4.7, 4.8 og 4.9 er sky-til-sky-distansene til henholdsvis Modell 4, Modell 2 og Modell 1 presentert. Her vises det at flatene målt på stålbjelkene får bedre georeferering i Modell 2 og 1, og det foreligger en forskjell på rundt 1 cm i RMSE i forhold til Modell 4. RMSE-verdiene viste tilnærmet lik kvalitet på georefereringen i Modell 1 og Modell 2. Det er viktig å nevne at kun to GCP-er ble synlig i Modell 2. Dette er et lavt antall. Da det ikke finnes detaljert beskrevet informasjon om SfM-metoden til Agisoft, er det ikke grunnlag for å si så mye om hvordan disse blir inkorporert i strålebuntutjevningen. Ved å se på RMSE-tabellene i sky-til-sky-analysen av Modell 1 og 2 er det lite som tilsier at de to GCP-ene forbedrer georefereringen til den endelige modellen. Det hadde vært hensiktsmessig å hatt flere GCP-er som var synlig i skråbildene, da modellen ikke kobler nadir-bildene og skråbildene slik som beskrevet i 5.1.1. Det var forventet at SfM-metoden ville klare å koble disse bildene. Dermed ble det ikke gjort tiltak for å få GCP-ene synlig i skråbildene. Dette burde blitt prioritert i feltarbeidet for å få et større innblikk i hvordan GCP-er påvirker georefereringen brumodellen.

5.2 Erfaringer ved praktisk gjennomføring

5.2.1 Flyplan

Flyplanen ga tilstrekkelig dekning på de fleste flatene på brua. Som diskutert i 5.1.1 var det ikke nødvendig å inkludere nadir-bildene i modellen. Endene av brua fikk ikke tilstrekkelig med bildevinkler, men der var årsaken fysiske hindringer i området.

H-bjelkene ble ikke fullstendig modellert grunnet manglende bildevinkler. Figur 5.2 viser flatene som det var vanskelig/umulig å få bilder av. Bunnen av alle H-bjelkene fikk derfor feilaktig geometri i alle modellene, med unntak av yttersiden av de ytterste H-bjelkene, ettersom det ble tatt bilder ovenifra ned på bjelkene. Det vil ikke være realistisk å få fullstendig modellering av bunnen av H-bjelkene. Dette hadde krevd at man fløy over bjelkene under brua, noe som er for risikabelt med tanke på kollisjonsfare.



Figur 5.2: Flater på H-bjelkene som ikke blir dekket av flyplanen.

Under flygingen ble det erfart at det var mulig å få RTK-signaler under denne brua. Uten GNSS-signal under brua begynte dronen å drifte, noe som gjorde det vanskelig å ta bilder. GNSS-fix ble oppnådd ved å fly så langt under brua som mulig. Det var kun fix i begrensede områder mellom hver pilar. Derfor var det også begrenset med bildevinkler som var mulig å få med av undersiden.

I endene var det ikke mulig å få fix, da det var for lavt under brua. Derfor ble det også for få bilder i disse områdene. En mulighet for å få fullstendig dekning i disse områdene kunne vært å bruke håndholdt kamera.

5.2.2 Bildetagning

Det homogene lyset fra det overskyede været ga optimale forhold for fotogrammetriske bilder. I tillegg var det snø under og rundt brua, noe som gjorde at det ble reflektert lys opp i de normalt mørkere områdene under brua. Det er viktig å understreke at det i realiteten ikke alltid vil være slike forhold under fotogrammetriprosjekter, og dermed er det ikke sikkert resultatet av denne flygingen ville vært den samme under andre forhold. Store deler av året er det ikke is og snø under de fleste bruer. Dette vil normalt gjøre det mørkere under brua, og vil mest sannsynlig ha utslag på modellene. Samtidig vil for eksempel dager med sterkt sollys kunne gjøre bildekvaliteten dårligere. Ved sollys kan det oppstå uønskede refleksjoner i flater, samt refleksflekker i bildene. Dette kompliserer også bildetagningen da det oppstår områder med skygger.

Det ble erfart at det ved en manuell flygning er vanskelig for en dronepilot å ta bilder og fly samtidig. Det mest ideelle er å justere kameraparametre underveis i flygingen, ved en ekstra person som endrer kamerainnstillinger og bestemmer bildevinkler med en ekstern skjerm.

I denne flygningen ble det brukt to kameraer av praktiske grunner. Resultatene viser at dette har gått fint, og at programvaren har klart å matche bildene. Det kan likevel tenkes at det vil være en fordel å bare bruke ett kamera for å sikre optimal bildematching.

5.3 Deteksjon av rust

I denne oppgaven har det blitt lagt frem en metode for å klassifisere og kvantifisere rust på punktsky og overflatemodell. Det har blitt benyttet to sett med terskelverdier for å teste om ulike kriterier gir en bedre klassifisering. Det har også blitt implementert en maskinlæringsmodell.

Figur 4.26 og 4.27 viser resultatet av de forskjellige klassifiseringene. Forskjellene i forsøkene viser at den strenge klassifiseringen som kun har brukt forholdstall mellom fargene har ført til at flere av de mørkere rustområdene har blitt detektert. Samtidig har mindre av rustavrenningene blitt klassifisert som rust. Klassifiseringen med Random Forest har klassifisert flest av de mørke rustområdene, men har også hatt

mange feilklassifiseringer i de mørkeste områdene på brua slik som vist i figur 4.32. Dette vises også i fargeplottene i figur 4.29 og 4.30. Dette kan være konsekvensen av at Random Forest-metoden har hatt en større andel mørke punkter i treningsdataen enn de andre metodene.

De fleste av de faktiske korrosjonsflekkene har lave RGB-verdier og framstår som mørke. Problemet her er at det forekommer mange mørke/svarte områder på stålbjelkene som ikke er rust. Det er derfor problematisk å skille ut det som faktisk er rust basert på kun RGB-verdier. Videre analyse av bildene tatt av disse problemområdene viser at det var mulig å få lysere bilder i disse områdene ved å gjøre en mer grundig jobb med justering av kameraparametre. Dette understreker det som ble diskutert i 5.2.2, og viser at det er viktig for dronepiloter å ha et markant fokus på bildetagning under flygning.

5.3.1 Forbedringer av deteksjonene

Deteksjon med maskinlæring ble gjort i siste delen av dette prosjektet. Grunnet tidsbegrensninger har en del eksperimentering med maskinlæring blitt utelatt. Optimalisering av hyperparametre, andre maskinlæringsteknikker, samt å se nærmere på bruk av ulike treningssett, kunne vært verdifullt for å optimalisere resultatet. Det er forventet at resultatet av rustdeteksjon basert på RGB-verdier vil være begrenset av de mørke områdene oppdaget i denne modellen.

5.3.2 Svakheter ved deteksjon ved rust på fotogrammetrimodell

En svakhet med rustdeteksjon på en fotogrammetrimodell er behovet for nøyaktig overensstemmelse mellom fargene i modellen og fargene i virkeligheten. I en fotogrammetrimodell vil ikke punkter nødvendigvis ha den fargen som de representerer i den virkelige verden. Terskelverdiene må settes ut i fra fargene på modellen. Hvis terskelverdiene baseres på farger som forventes å være til stede i den virkelige verden, kan det resultere i feildeteksjoner. I tillegg kan lysforhold, som for eksempel skygger eller sollys, endre fargegjengivelsen og føre til variasjoner på modellen. Det vil si at to områder med like farger i virkeligheten kan få forskjellige farger i modellen.

En annen svakhet ved disse metodene er at det bare blir analysert ett punkt om gangen. Dette gjør at det ikke blir tatt hensyn til hvordan punktene relaterer til hverandre. Dermed blir egenskaper som form og tekstur neglisjert.

Disse metodene forutsetter også at fargeverdiene til rust skiller seg noe ut fra fargen på metallflaten der det forsøkes å klassifisere rust. Rust har som regel en særegen farge, men hvis stålbjelkene på en bru har for like farger, kan dette gjøre en deteksjon basert på RGB-verdier vanskeligere.

5.4 Videre arbeid

5.4.1 Implementering av multispektrale bilder i modellering

RGB-verdiene i 3D-modellen gir begrenset informasjon om punktene. Dermed blir også beslutningsgrunnlaget for å klassifisere rust automatisk innskrenket. En metode for å gjøre dette beslutningsgrunnlaget større kunne vært å teste multispektrale bilder. Et multispektralt bilde inneholder flere bånd og omfatter en større del av det elektromagnetiske spekteret. Fotogrammetri med multispektrale bilder er mulig med kommersiell programvare, og kan produsere punktskyer der punktene har flere attributter enn bare rød, grønn og blå.

5.4.2 Automatisk segmentering av bjelkene

Det ble forsøkt å segmentere H-bjelkene ved hjelp av RANSAC-algoritmen. Det ble testet forskjellige terskelverdier, men det var ikke mulig å isolere objektene på brua med stor nok presisjon.

For segmentering av kompliserte geometriske objekter viser dyp læring lovende resultater. PointNet er et nevralt nettverk som er laget for implementering på punktskyer, og som trenes for å lære seg kjennetegn på geometriske objekter. Dette kunne vært interessant å teste på bruer.

Kapittel 6

Konklusjon

Introduksjonen til oppgaven tok for seg følgende spørsmål:

1. Hvordan bør det tas bilder av en bru hvis den skal modelleres i en fotogrammetriprosess, og hva slags kvalitet kan oppnås i en slik modell?
2. Kan man klassifisere og kvantifisere korrosjon på stålbjelkene ved hjelp av en fotogrammetrimodell?

For å svare på disse spørsmålene ble det utarbeidet et forslag til en flyplan tilpasset fotogrammetriske formål. Det har blitt laget modeller som det har blitt gjort diverse analyser på. Til slutt ble det undersøkt om det var mulig å bruke modellene til å klassifisere og kvantifisere korrosjon.

Det overordnede resultatet fra alle modellene viser at det er vanskelig å få en komplett digital gjenskapning av en bru basert på fotogrammetri. Flygningen har blitt gjort etter en teoretisk flyplan utarbeidet spesielt med tanke på tredimensjonal rekonstruksjon, og har fungert for områdene der det ikke var hindringer. Det har blitt erfart at spesielt stålbjelkene består av mangler og geometriske forvrengninger i modellene. Det har blitt konkludert med at de homogene flatene på stålbjelkene er hovedårsaken til ufullstendigheten på modellene. Samtidig er det erfart at det er vanskelig å få tilstrekkelig med bildevinkler under brua, og at dette er et problem som sannsynligvis er overførbart til andre bruer. Det er tenkelig at en modell kan bli brukt som et supplement til en vanlig bruinspeksjon, men detaljnivået på modellen er ikke bra nok til å utføre inspeksjonen på bjelkene i modellen. I modellene blir betongflatene mest komplette. Derfor hadde det vært interessant å videre undersøke potensialbruken av disse i bruinspeksjonsdelen.

Det er vist at flyplanen er en effektiv måte å samle inn data av en bru på. Den store mengden bilder som samles inn dekker hele brua og gir en god indikasjon på tilstanden. Det er tenkelig at det over tid kan utføres flere flygninger som på sikt kan gi en god indikasjon på endring i tilstand. Samtidig blir brua georeferert i globale koordinater på centimeternivå.

Rustdeteksjonen viste at det var mulig å gjøre arealregninger på korrodert område. Selv om stålbjelkene får mange ufullstendige flater i modellen er rustflekkene godt

modellert. Dermed kan det gjøres arealberegninger med god nøyaktighet. Metoden brukt i denne oppgaven gir ikke et nøyaktig areal på korrodert stål, men klarer fint å gi en visuell indikasjon på hvor det er mye rust. De mørke områdene på modellen viser seg å være de mest problematiske områdene, da det ligger begrenset med informasjon i RGB-verdiene.

Bibliografi

B. Dick, Øystein (2020a). *Fotogrammetri*. URL: https://snl.no/fotogrammetri. (accessed: 24.01.2023).

Andersen, Øystein (2003). *Orientering i stereoinstrument*. Institutt for kartfag, NLH.

B. Dick, Øystein (2020b). *Sentralprojeksjon*. URL: https://snl.no/sentralprojeksjonm. (accessed: 22.02.2023).

Luhmann, Thomas, Stuart Robson, Stephen Kyle og Jan Boehm (2020). *Close-Range Photogrammetry and 3D Imaging*. Walter de Gruyter GmbH. ISBN: 978-3-11-060724-6.

Lowe, David (nov. 2004). "Distinctive Image Features from Scale-Invariant Keypoints". I: *International Journal of Computer Vision* 60, s. 91–. DOI: 10.1023/B:VISI.0000029664.99615.94.

Triggs, Bill, Philip Mclauchlan, Richard Hartley og Andrew Fitzgibbon (sep. 2000). "Bundle Adjustment - A Modern Synthesis". I: *International Workshop on Vision Algorithms*. Red. av Bill Triggs, Andrew Zisserman og Richard Szeliski. Bd. 1883. Lecture Notes in Computer Science. Springer-Verlag, s. 298–372. DOI: 10.1007/3-540-44480-7_21. URL: https://inria.hal.science/inria-00548290.

Skogseth, Terje og Dag Norberg (2014). *Grunnlegende Landmåling*. Gyldendal Norsk Forlag As. ISBN: 978-82-05-44934-3.

Teunissem, Peter J.G og Oliver Montenbruck (2017). *Springer Handbook of Global Navigation Satellite Systems*. Springer Handbooks. Springer International Publishing AG. ISBN: 978-3-030-73172-4.

O'Connor, James, Mike J Smith og Mike R James (2017). "Cameras and settings for aerial surveys in the geosciences: Optimising image data". I: *Progress in Physical Geography: Earth and Environment* 41.3, s. 325–344. DOI: 10.1177/0309133317703092. URL: https://doi.org/10.1177/0309133317703092.

Wang, Feng, Yang Zou, Enrique del Rey Castillo, Youliang Ding, Zhao Xu, Hanwei Zhao og James B.P. Lim (2022). "Automated UAV path-planning for high-quality photogrammetric 3D bridge reconstruction". I: *Structure and Infrastructure Engineering* 0.0, s. 1–20. DOI: 10.1080/15732479.2022.2152840. URL: https://doi.org/10.1080/15732479.2022.2152840.

Pan, Yue, Yiqing Dong, Dalei Wang, Airong Chen og Zhen Ye (2019). "Three-Dimensional Reconstruction of Structural Surface Model of Heritage Bridges Using UAV-Based Photogrammetric Point Clouds". I: *Remote Sensing* 11.10. ISSN: 2072-

4292. DOI: 10.3390/rs11101204. URL: https://www.mdpi.com/2072-4292/11/10/1204.

Raschka, S. og V. Mirjalili (2019). *Python Machine Learning - Third Edition*. Packt Publishing. ISBN: 978-1-78995-575-0.

Tillegg A

Tabeller

Tabell A.1: Oversikt over punktene som ble kategorisert som rust og tilhørende RGB-verdier ved de milde terskelverdiene.

Rød	Grønn	Blå	Rød/Grønn	Rød/Blå	Grønn/Blå
200	184	140	1,09	1,43	1,31
78	47	25	1,66	3,12	1,88
101	62	41	1,63	2,46	1,51
165	137	74	1,20	2,23	1,85
92	38	22	2,42	4,18	1,73
71	35	27	2,03	2,63	1,30
99	61	41	1,62	2,41	1,49
92	69	38	1,33	2,42	1,82
94	40	27	2,35	3,48	1,48
109	70	41	1,56	2,66	1,71
96	85	68	1,13	1,41	1,25
87	46	38	1,89	2,29	1,21
121	78	51	1,55	2,37	1,53
141	94	56	1,50	2,52	1,68
134	106	78	1,26	1,72	1,36
173	151	118	1,15	1,47	1,28
154	118	69	1,31	2,23	1,71
93	67	62	1,39	1,50	1,08
111	66	47	1,68	2,36	1,40
76	50	43	1,52	1,77	1,16
142	110	78	1,29	1,82	1,41
156	132	109	1,18	1,43	1,21

Tabell A.2: Minste og største verdiene i datasettet ved de milde terskelverdiene

	Rød	Grønn	Blå	Rød/Grønn	Rød/Blå	Grønn/Blå
Minste verdi	71	35	22	1,09	1,41	1,08
Høyeste verdi	200	184	140	2,42	4,18	1,88

Tabell A.3: Oversikt over punktene som ble kategorisert som rust og tilhørende RGB-verdier ved de strenge terskelverdiene.

Rød	Grønn	Blå	Rød/Grønn	Rød/Blå	Grønn/Blå
79	27	15	2,93	5,27	1,80
38	22	18	1,73	2,11	1,22
42	29	20	1,45	2,10	1,45
80	35	15	2,29	5,33	2,33
123	66	43	1,86	2,86	1,53
77	24	14	3,21	5,50	1,71
96	38	19	2,53	5,05	2,00
36	16	12	2,25	3,00	1,33
73	37	26	1,97	2,81	1,42
80	31	20	2,58	4,00	1,55
48	30	26	1,60	1,85	1,15
95	57	47	1,67	2,02	1,21
110	53	25	2,08	4,40	2,12
126	69	47	1,83	2,68	1,47
111	65	39	1,71	2,85	1,67
71	40	26	1,78	2,73	1,54
64	41	31	1,56	2,06	1,32
61	32	17	1,91	3,59	1,88
71	38	36	1,87	1,97	1,06
106	57	42	1,86	2,52	1,36

Tabell A.4: Minste og største verdiene i datasettet ved de strenge terskelverdiene

	Rød	Grønn	Blå	Rød/Grønn	Rød/Blå	Grønn/Blå
Minste verdi	36	16	12	1,45	1,85	1,06
Høyeste verdi	126	69	47	3,21	5,50	2,33

Tillegg B

Pythonkode

B.1 Transformasjon mellom koordinatsystemer

```python
# -*- coding: utf-8 -*-

import pyproj
import laspy
import numpy as np

# Input LAS file path
innfil = "D:/Ulvesund bru/Scan med gcp/BraLidarScan/cloud-12876ce0.las"

# Read LAS file using laspy
infile = laspy.read(innfil)

# Stack point cloud data (X, Y, Z, R, G, B)
pc = np.stack((infile.x, infile.y, infile.z, infile.red, infile.green, infile.blue), -1)

def transformation(crs_from=4937, crs_to=5972):
    """
    Transform coordinates from one CRS to another.

    Parameters
    ----------
    crs_from : int, optional
        Source coordinate reference system (CRS) code, by default 4937.
    crs_to : int, optional
        Target coordinate reference system (CRS) code, by default 5972.

    Returns
    -------
    Transformer, float
        Coordinate transformer object and current epoch.
    """
    # https://epsg.io/7912 EPSG 7912 is ITRF14, lat lon, height
    # https://epsg.io/5972 ETRS89 / UTM zone 32N + NN2000 height
```

```
    # https://epsg.io/4937 ETRS89 with lat lon ellipsoid, lidarscannene våre kommer ut som denne, ikke WGS84!!!!

    from pyproj import Transformer
    import pandas as pd

    dayofyear = pd.Period("2023-02-14", freq="H").day_of_year
    currentepoch = int(2023) + int(dayofyear) / 365  # Current Epoch ex: 2021.45
    return Transformer.from_crs(crs_from, crs_to), currentepoch

# Create transformer object and get current epoch
transformer, current_epoch = transformation()

# Transform point cloud coordinates to target CRS
New_Pointcloud = []
for y, x, z, r, g, b in pc:
    X_Euref89, Y_Euref89, Z_Euref89, epoch_init = transformer.transform(x, y, z, current_epoch)
    New_Pointcloud.append([X_Euref89, Y_Euref89, Z_Euref89, r, g, b])

def draw_las(data, file_name, epsg):
    """
    Save transformed point cloud data as a new LAS file.

    Parameters
    ----------
    data : list
        List of transformed point cloud data.
    file_name : str
        Output file name.
    epsg : int
        EPSG code for the target coordinate reference system.
    """

    # Convert data to a NumPy array
    my_data = np.array(data)

    # Create LAS header with point format 3 and version 1.2
    header = laspy.LasHeader(point_format=3, version="1.2")
    header.add_extra_dim(laspy.ExtraBytesParams(name="random", type=np.int32))
    header.offsets = np.array([np.min(my_data[:, 0]), np.min(my_data[:, 1]), np.min(my_data[:, 2])])
    header.scales = np.array([0.001, 0.001, 0.001])
    header.crs = header.add_crs(pyproj.CRS(epsg))

    # Create a new LAS file with the specified header
    las = laspy.LasData(header)

    # Assign point cloud data to the LAS file
    las.x = my_data[:, 0]
    las.y = my_data[:, 1]
    las.z = my_data[:, 2]
    las.red = my_data[:, 3]
    las.green = my_data[:, 4]
    las.blue = my_data[:, 5]
```

```
86
87      # Write the LAS file to disk
88      las.write(file_name + ".las")
89
90  # Save the transformed points
91  draw_las(New_Pointcloud, "D:/Testfiler/Rust_Red_Values", epsg=5972)
```

B.2 Rustdeteksjon Punktsky

```
# -*- coding: utf-8 -*-

import numpy as np
import laspy
import pyproj


# Input LAS file path
innfil = "D:/Ferdige modeller/Uten bildene over/Deler/H-BjelkerSegmentert.las"

# Read LAS file using laspy
infile = laspy.read(innfil)

# Stack point cloud data (X, Y, Z, R, G, B)
colours = np.stack((infile.x, infile.y, infile.z, infile.red, infile.green, infile.blue), -1)

Rust_Red_Values = []     #Detected rust in red
Rust_points = []         #Just detected rust


# Filter point cloud data based on color values
for x, y, z, r, g, b in colours:
    if (r/g > 1.45) & (r/b > 1.85) & (g/b > 1.15):
# if ((70/255*65536) < r < (200/255*65536))  ((30/255*65536) < g < (185/255*65536))  (b < (140/255*65536))  (r/g > 1.09)  (r/b > 1.4)  (g/b > 1.15):
        Rust_points.append([x, y, z, r, g, b])
        Rust_Red_Values.append([x, y, z, 65535, 0, 0])
    else:
        Rust_Red_Values.append([x, y, z, r, g, b])

def draw_las(data, file_name, epsg):
    """
    Save filtered point cloud data as a new LAS file.

    Parameters
    ----------
    data : list
        List of filtered point cloud data.
    file_name : str
        Output file name.
    epsg : int
        EPSG code for the target coordinate reference system.
    """

    # Convert data to a NumPy array
    my_data = np.array(data)

    # Create LAS header with point format 2 and version 1.2
    header = laspy.LasHeader(point_format=2, version="1.2")
    header.add_extra_dim(laspy.ExtraBytesParams(name="random", type=np.int32))
    header.offsets = np.array([np.min(my_data[:, 0]), np.min(
```

```
    my_data[:, 1]), np.min(my_data[:, 2])])
51      header.scales = np.array([0.001, 0.001, 0.001])
52      header.crs = header.add_crs(pyproj.CRS(epsg))
53
54      # Create a new LAS file with the specified header
55      las = laspy.LasData(header)
56
57      # Assign point cloud data to the LAS file
58      las.x = my_data[:, 0]
59      las.y = my_data[:, 1]
60      las.z = my_data[:, 2]
61      las.red = my_data[:, 3]
62      las.green = my_data[:, 4]
63      las.blue = my_data[:, 5]
64
65      # Write the LAS file to disk
66      las.write(file_name + ".las")
67
68  # Save the filtered point cloud as new LAS files
69  draw_las(Rust_Red_Values, "D:/Testfiler/Rust_Red_Values", epsg
        =5972)
70  draw_las(Rust_points, "D:/Testfiler/Rust_points", epsg=5972)
```

B.3 Rustdeteksjon Mesh

```
# -*- coding: utf-8 -*-

import open3d as o3d
import numpy as np

# Load OBJ file
mesh_path = "D:/Ferdige modeller/UtenBilderOverUtenGCP/Mesh/
    BareBjelker.obj"
mesh = o3d.io.read_triangle_mesh(mesh_path)

colors = np.asarray(mesh.vertex_colors)
triangles = np.array(mesh.triangles)
vertices = np.asarray(mesh.vertices)


"""
# Define the color criterion for rust detection
r_min, r_max = 70 / 255, 200 / 255
g_min, g_max = 30 / 255, 185 / 255
b_max = 140 / 255


matching_indices = np.where(
    (colors[:, 0] > r_min) & (colors[:, 0] < r_max) &
    (colors[:, 1] > g_min) & (colors[:, 1] < g_max) &
    (colors[:, 2] < b_max) &
    (colors[:, 0] / colors[:, 1] > 1.09) &
    (colors[:, 0] / colors[:, 2] > 1.4) &
    (colors[:, 1] / colors[:, 2] > 1.15))[0]
"""

matching_indices = np.where(
    (colors[:, 0] / colors[:, 1] > 1.45) &
    (colors[:, 0] / colors[:, 2] > 1.85) &
    (colors[:, 1] / colors[:, 2] > 1.15)
)[0]

matching_triangles = mesh.select_by_index(matching_indices)

# calculate area

Area_mesh = mesh.get_surface_area()
Area_Rust = matching_triangles.get_surface_area()


print(f"Total area of input mesh: {Area_mesh:.4f}")
print(f"Total area of rust triangles: {Area_Rust:.4f}")

# Create a new mesh with the same vertices and triangles as the original mesh
new_mesh = mesh
red_color = [1.0, 0.0, 0.0]

new_vertex_colors = np.array(new_mesh.vertex_colors)

for i, tri_idx in enumerate(new_mesh.triangles):
```

```
55      if i in matching_indices:
56          new_mesh.vertex_colors[i] = red_color
57
58  o3d.visualization.draw_geometries([new_mesh])
59  o3d.visualization.draw_geometries([mesh])
60
61  output_path = "D:/Testbilder/MeshLocal_coloredHard.ply"
62  o3d.io.write_triangle_mesh(output_path, new_mesh)
```

B.4 Rustdeteksjon Random Forrest

```
# -*- coding: utf-8 -*-

# Import necessary libraries
import numpy as np
import pyproj
import laspy
import open3d as o3d
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from tqdm import tqdm

# Define the file paths
Rust_path="D:/Ferdige modeller/H-BjelkerRust.las"
NonRust_path="D:/Ferdige modeller/H-BjelkerIKKErust2.las"
input_file_path="D:/Ferdige modeller/Uten bildene over/Deler/H-
    BjelkerSegmentert.las"

# Read the .las files
input_file = laspy.read(input_file_path)
NonRust = laspy.read(NonRust_path)
Rust = laspy.read(Rust_path)

# Stack the data
colours = np.stack((input_file.x, input_file.y, input_file.z,
    input_file.red, input_file.green, input_file.blue),-1)
NonRust = np.stack((NonRust.red, NonRust.green, NonRust.blue),-1)
Rust = np.stack((Rust.red, Rust.green, Rust.blue),-1)

# Prepare the test data
Testdata = colours[:,3:6]/(2**16)*255

# Prepare the training data
Rust = Rust[:,0:3]/(2**16)*255
Rust_labels = np.ones(len(Rust))
Rust_train = np.column_stack((Rust, Rust_labels))

# Prepare the training data
NonRust = NonRust[:,0:3]/(2**16)*255
NonRust_labels = np.zeros(len(NonRust))
NonRust_train = np.column_stack((NonRust, NonRust_labels))

# Concatenate the data
data = np.concatenate((NonRust_train,Rust_train))

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(data[:, 0:3],
    data[:, -1], test_size=0.3, random_state=42)

# Initialize a random forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier on the training data
clf.fit(X_train, y_train)

```

```
# Evaluate the classifier on the test data
accuracy = clf.score(X_test, y_test)
print("Test accuracy: {:.2f}%".format(accuracy * 100))

# Make predictions on the new data
predictions = clf.predict(Testdata[:,0:3])

# Reshape the predictions and merge with the original data
predictions_reshaped = predictions.reshape(-1, 1)
merged_data = np.column_stack((colours[:,0:3], Testdata,
    predictions_reshaped))

# Print the predicted labels for each data point
print(predictions)

# Filter points classified as rust
classified_points = merged_data[predictions == 1]

# Create a boolean mask where the last column has a value of 1
mask = merged_data[:, 6] == 1

# Replace the RGB values in the data with red where the mask is True
merged_data[mask, 3] = 255
merged_data[mask, 4] = 0
merged_data[mask, 5] = 0

# Function to draw .las files
def draw_las(data, file_name, epsg):
    my_data = np.array(data)
    header = laspy.LasHeader(point_format=2, version="1.2")
    header.add_extra_dim(laspy.ExtraBytesParams(name="random", type
    =np.int32))
    header.offsets = np.array([np.min(my_data[:, 0]), np.min(
    my_data[:, 1]), np.min(my_data[:, 2])])
    header.scales = np.array([0.001, 0.001, 0.001])
    header.crs = header.add_crs(pyproj.CRS(epsg))

    # Create a Las
    las = laspy.LasData(header)

    las.x = my_data[:, 0]
    las.y = my_data[:, 1]
    las.z = my_data[:, 2]
    las.red = my_data[:, 3]
    las.green = my_data[:, 4]
    las.blue = my_data[:, 5]

    las.write(file_name + ".las")

# Load OBJ file
mesh = o3d.io.read_triangle_mesh("D:/Ferdige modeller/
    UtenBilderOverUtenGCP/Mesh/BareBjelker.obj")

colors255 = np.asarray(mesh.vertex_colors) * 255
colors = np.asarray(mesh.vertex_colors)
triangles = np.array(mesh.triangles)
vertices = np.asarray(mesh.vertices)

```

```
107 predictions = clf.predict(colors255[:,0:3])
108 indices_array = np.where(predictions == 1)
109 indices_list = list(map(int, indices_array[0]))
110
111 matching_triangles = mesh.select_by_index(indices_list)
112
113 Area_mesh = mesh.get_surface_area()
114 Area_Rust = matching_triangles.get_surface_area()
115
116 o3d.visualization.draw_geometries([matching_triangles])
117
118 print("Total area of total input mesh: {:.4f}".format(Area_mesh))
119 print("Total area of rust triangles: {:.4f}".format(Area_Rust))
120
121 # Create a new mesh with the same vertices and triangles as the original mesh
122 new_mesh = mesh
123
124 # Change the color of triangles with matching indices to red
125 red_color = [1.0, 0.0, 0.0]
126 new_vertex_colors = np.array(new_mesh.vertex_colors)
127
128 # Updating vertex colors
129 for i, tri_idx in enumerate(tqdm(new_mesh.vertex_colors, desc="
      Updating vertex colors")):
130     if i in indices_list:
131         new_mesh.vertex_colors[i] = red_color
132
133 o3d.visualization.draw_geometries([new_mesh])
134 o3d.visualization.draw_geometries([mesh])
135
136 #
```

Tillegg C

Agisoft Metasape Modell 1 rapport

Survey Data

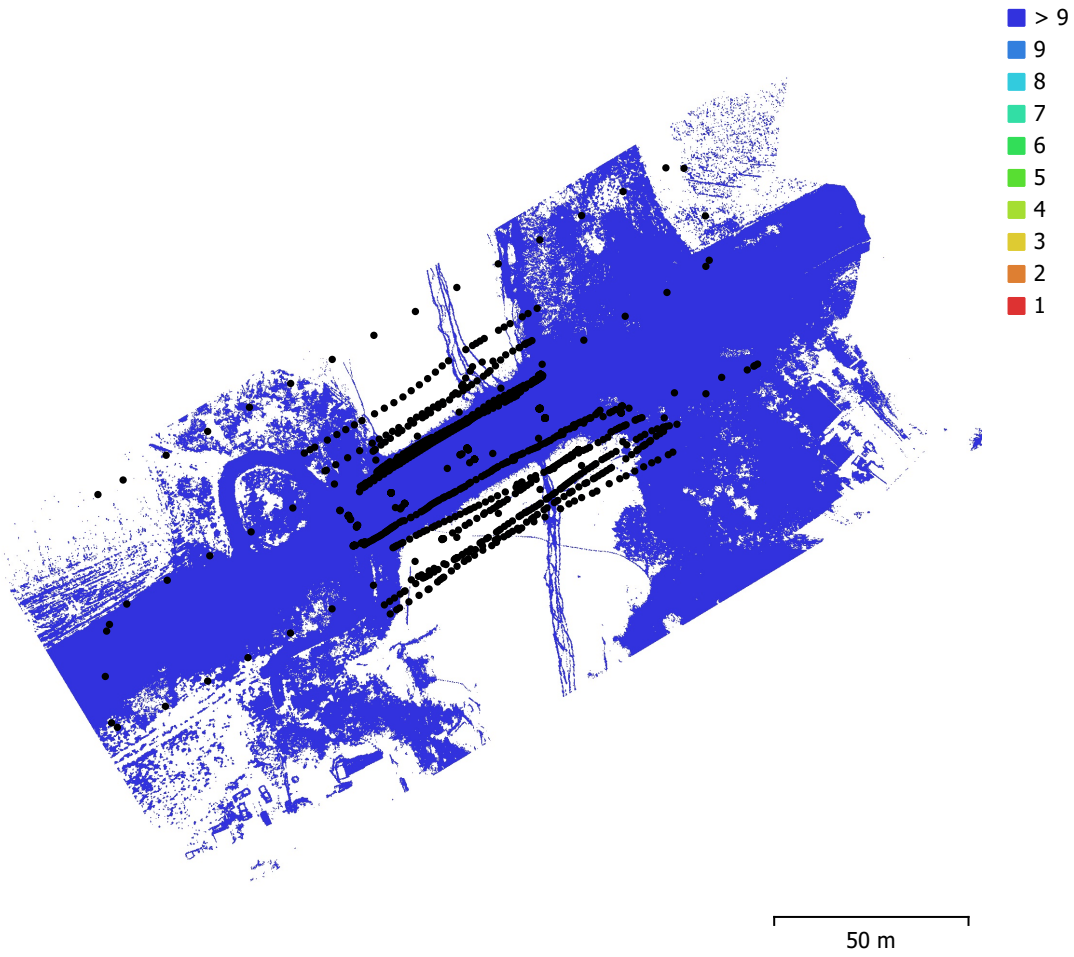


Fig. 1. Camera locations and image overlap.

Number of images:	984	Camera stations:	984
Flying altitude:	11.6 m	Tie points:	987,094
Ground resolution:	3.31 mm/pix	Projections:	3,806,125
Coverage area:	0.014 km²	Reprojection error:	0.743 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
ZenmuseP1 (35mm)	8192 x 5460	35 mm	4.39 x 4.39 µm	No
ZH20 (4.5mm)	4056 x 3040	4.5 mm	1.6 x 1.6 µm	No
ZH20 (25.39mm)	5184 x 3888	25.39 mm	1.45 x 1.45 µm	No

Table 1. Cameras.

Camera Calibration

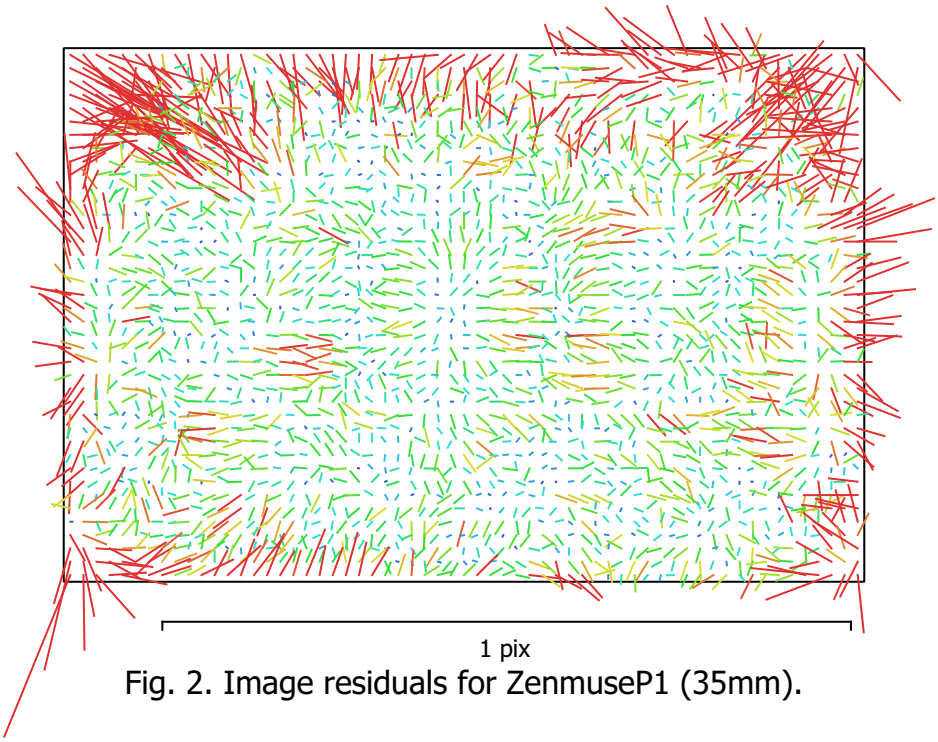


Fig. 2. Image residuals for ZenmuseP1 (35mm).

ZenmuseP1 (35mm)

647 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**8192 x 5460**	**35 mm**	**4.39 x 4.39 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**8190.07**	0.2	1.00	0.03	0.03	-0.98	0.96	-0.93	0.90	0.02	0.02
Cx	**-7.13066**	0.095		1.00	0.00	-0.02	0.02	-0.02	0.01	0.96	-0.00
Cy	**6.88497**	0.083			1.00	-0.03	0.03	-0.03	0.03	0.00	0.95
K1	**-0.0375319**	0.00044				1.00	-0.99	0.98	-0.95	-0.02	-0.02
K2	**0.0877116**	0.0029					1.00	-0.99	0.97	0.02	0.03
K3	**-0.320111**	0.0081						1.00	-0.99	-0.01	-0.03
K4	**0.273604**	0.0084							1.00	0.01	0.03
P1	**-0.000306058**	4.9e-06								1.00	0.00
P2	**0.00168939**	4.5e-06									1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Calibration

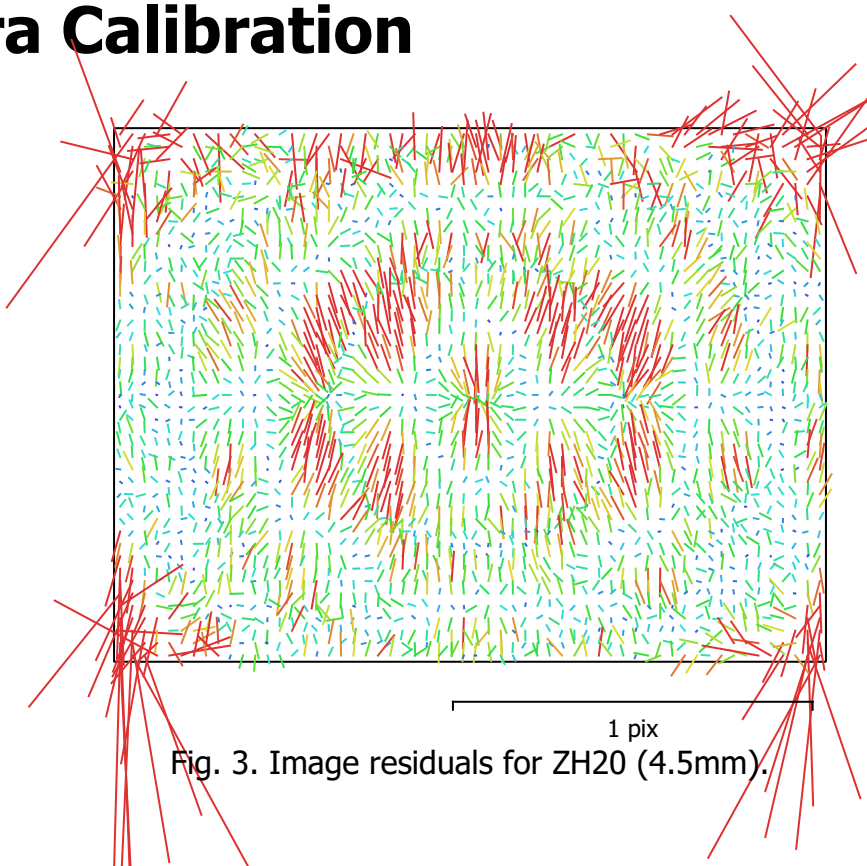


Fig. 3. Image residuals for ZH20 (4.5mm).

ZH20 (4.5mm)

328 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**4056 x 3040**	**4.5 mm**	**1.6 x 1.6 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**2987**	0.71	1.00	-0.06	0.05	-0.99	0.98	-0.95	0.91	-0.07	0.01
Cx	**35.8506**	0.11		1.00	0.01	0.06	-0.07	0.07	-0.07	0.92	0.04
Cy	**21.9244**	0.09			1.00	-0.03	0.03	-0.03	0.03	0.01	0.58
K1	**0.0327075**	0.0022				1.00	-0.99	0.98	-0.94	0.07	-0.02
K2	**0.018253**	0.0076					1.00	-0.99	0.97	-0.08	0.02
K3	**-0.561563**	0.012						1.00	-0.99	0.08	-0.02
K4	**0.555617**	0.0067							1.00	-0.08	0.01
P1	**7.97983e-05**	1.3e-05								1.00	0.03
P2	**0.000180061**	8.7e-06									1.00

Table 3. Calibration coefficients and correlation matrix.

Camera Calibration

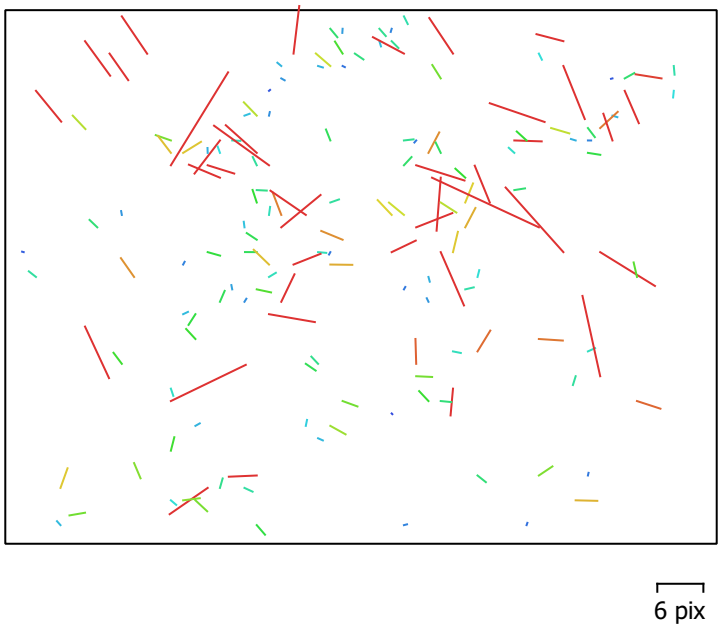


Fig. 4. Image residuals for ZH20 (25.39mm).

ZH20 (25.39mm)

9 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**5184 x 3888**	**25.39 mm**	**1.45 x 1.45 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**18033.4**	5.5e+02	1.00	-0.60	-0.63	-0.38	0.69	-0.84	0.79	-0.28	-0.04
Cx	**1843.96**	2.6e+03		1.00	0.91	-0.46	-0.06	0.65	-0.85	0.19	-0.17
Cy	**3305.61**	2.6e+03			1.00	-0.37	-0.23	0.78	-0.93	-0.11	-0.44
K1	**0.191198**	1.6				1.00	-0.79	0.26	0.04	0.09	0.23
K2	**5.65622**	34					1.00	-0.78	0.54	0.12	0.18
K3	**-108.94**	4.5e+02						1.00	-0.95	-0.08	-0.34
K4	**451.381**	2.2e+03							1.00	0.03	0.36
P1	**0.0185492**	0.018								1.00	0.90
P2	**0.0376491**	0.024									1.00

Table 4. Calibration coefficients and correlation matrix.

Camera Locations

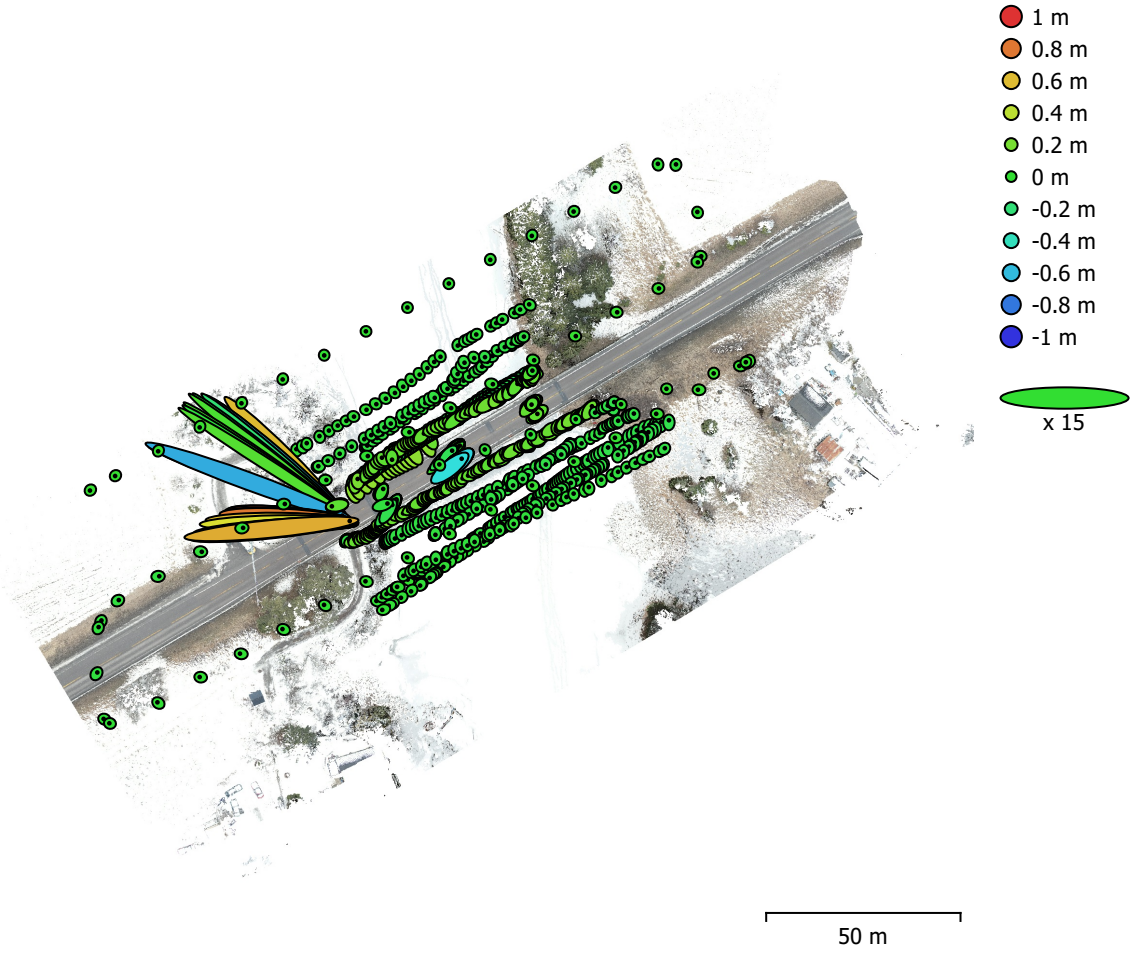


Fig. 5. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
38.7834	20.5367	13.7649	43.8852	45.9933

Table 5. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Digital Elevation Model

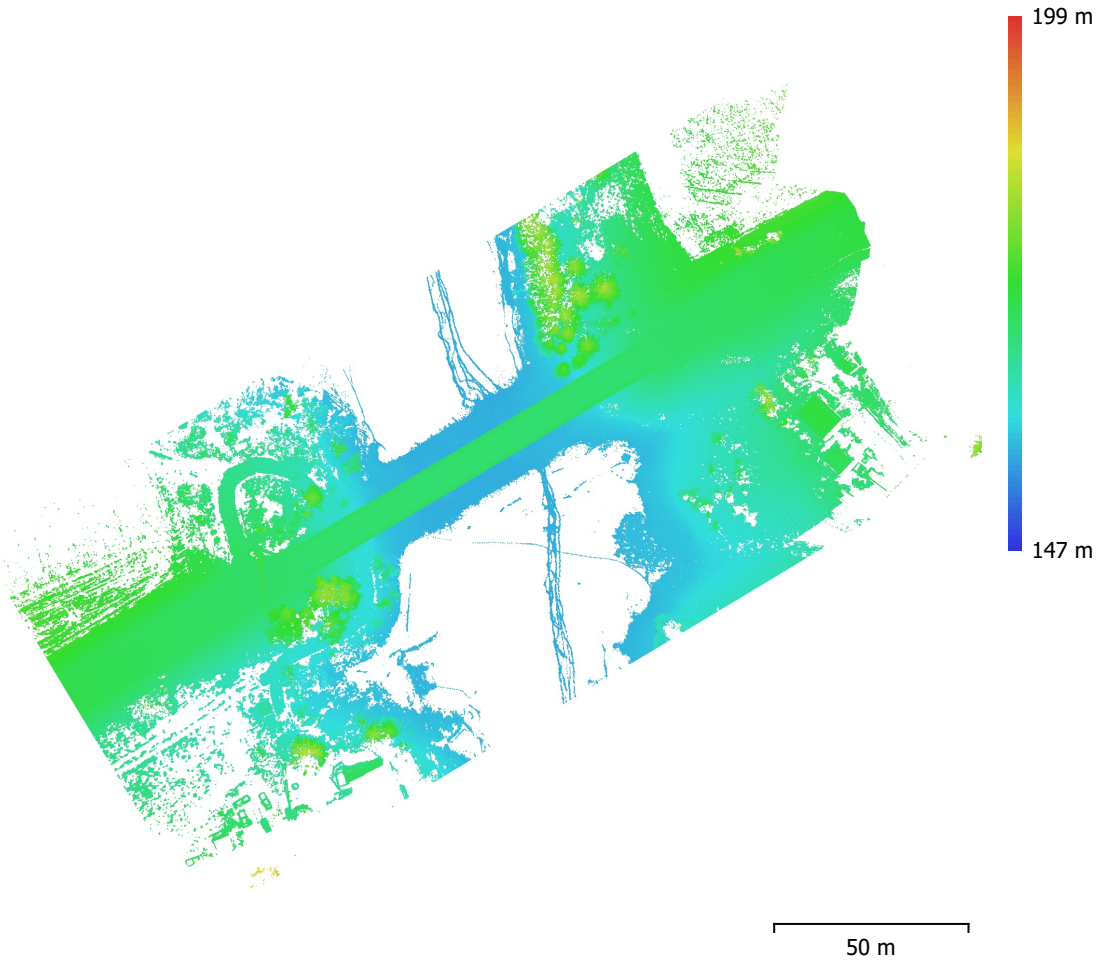


Fig. 6. Reconstructed digital elevation model.

Resolution:	unknown
Point density:	unknown

Processing Parameters

General	
Cameras	984
Aligned cameras	984
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	987,094 of 2,285,702
RMS reprojection error	0.142935 (0.742903 pix)
Max reprojection error	4.02855 (46.7076 pix)
Mean key point size	4.27494 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	4.00941
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Source
Key point limit	45,000
Key point limit per Mpx	1,000
Tie point limit	10,000
Exclude stationary tie points	No
Guided image matching	No
Adaptive camera model fitting	No
Matching time	31 minutes 22 seconds
Matching memory usage	3.73 GB
Alignment time	29 minutes 34 seconds
Alignment memory usage	788.29 MB
Optimization parameters	
Parameters	f, cx, cy, k1-k4, p1, p2
Fit additional corrections	Yes
Adaptive camera model fitting	No
Optimization time	48 minutes 45 seconds
Date created	2023:01:17 23:35:39
Software version	2.0.1.15925
File size	169.10 MB
Depth Maps	
Count	976
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 28 minutes
Memory usage	14.12 GB
Date created	2023:01:18 06:30:16
Software version	2.0.1.15925
File size	7.32 GB
Point Cloud	
Points	496,740,678
Point attributes	
Position	

Color	3 bands, uint8
Normal	
Confidence	
Point classes	
Created (never classified)	496,740,678
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 28 minutes
Memory usage	14.12 GB
Point cloud generation parameters	
Processing time	7 hours 14 minutes
Memory usage	25.59 GB
Date created	2023:01:18 13:45:09
Software version	2.0.1.15925
File size	7.08 GB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.1 build 15925
OS	Windows 64 bit
RAM	31.76 GB
CPU	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
GPU(s)	Intel(R) UHD Graphics Quadro T1000

Tillegg D

Agisoft Metasape Modell 2 rapport

Survey Data

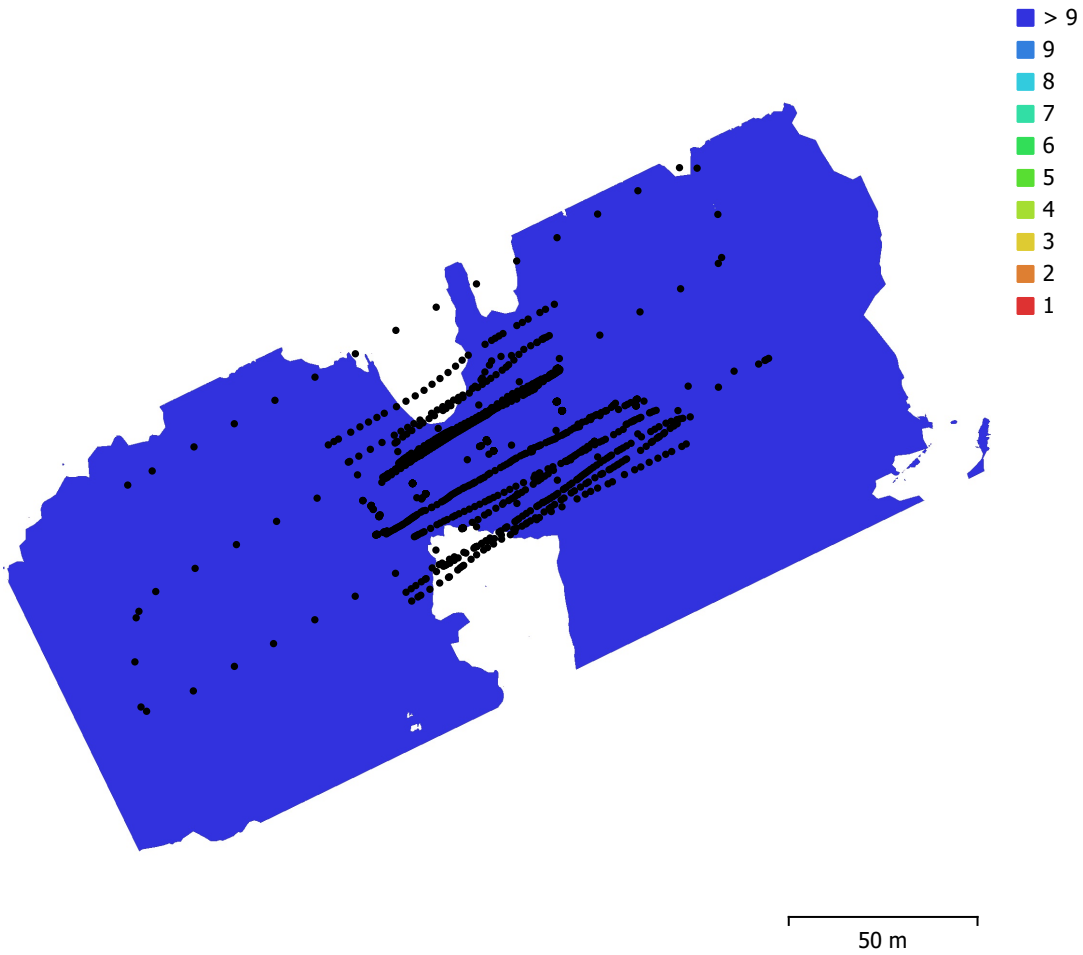


Fig. 1. Camera locations and image overlap.

Number of images:	984	Camera stations:	984
Flying altitude:	11.6 m	Tie points:	1,310,307
Ground resolution:	3.32 mm/pix	Projections:	5,252,598
Coverage area:	0.0234 km²	Reprojection error:	0.769 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
ZenmuseP1 (35mm)	8192 x 5460	35 mm	4.39 x 4.39 µm	No
ZH20 (4.5mm)	4056 x 3040	4.5 mm	1.6 x 1.6 µm	No
ZH20 (25.39mm)	5184 x 3888	25.39 mm	1.45 x 1.45 µm	No

Table 1. Cameras.

Camera Calibration

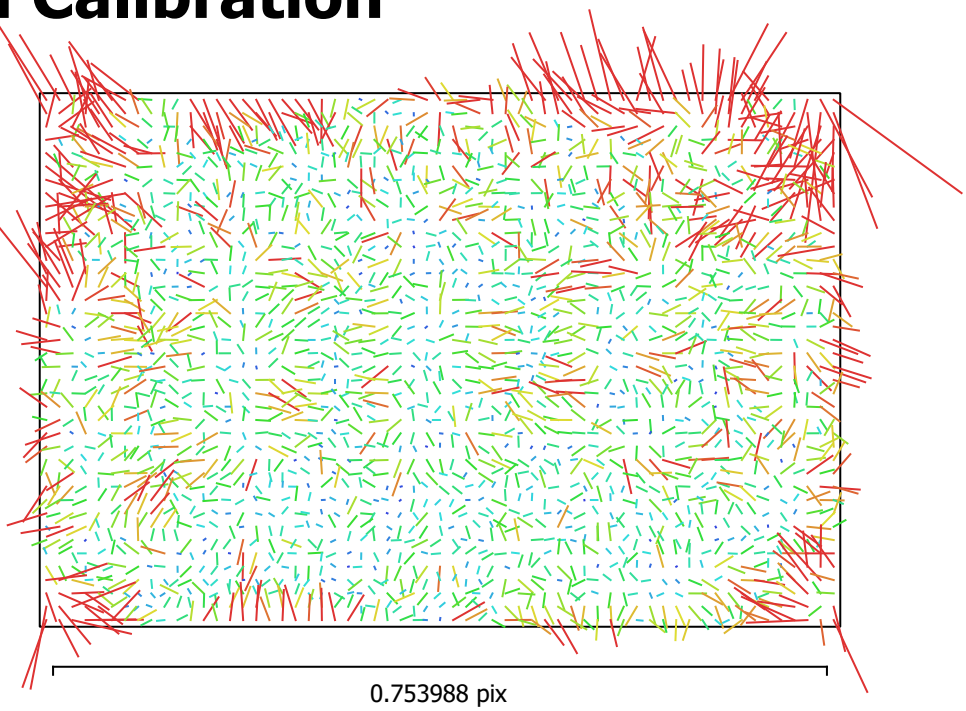


Fig. 2. Image residuals for ZenmuseP1 (35mm).

ZenmuseP1 (35mm)

647 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**8192 x 5460**	**35 mm**	**4.39 x 4.39 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**8189.4**	0.22	1.00	0.03	0.03	-0.98	0.96	-0.93	0.90	0.02	0.02
Cx	**-8.35091**	0.1		1.00	0.01	-0.03	0.02	-0.02	0.01	0.96	0.00
Cy	**8.18092**	0.088			1.00	-0.03	0.03	-0.03	0.03	0.00	0.95
K1	**-0.036755**	0.00047				1.00	-0.99	0.97	-0.94	-0.02	-0.03
K2	**0.0816143**	0.0031					1.00	-0.99	0.97	0.02	0.03
K3	**-0.299446**	0.0087						1.00	-0.99	-0.02	-0.03
K4	**0.244082**	0.009							1.00	0.01	0.03
P1	**-0.000395797**	5.3e-06								1.00	0.00
P2	**0.00174742**	4.8e-06									1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Calibration

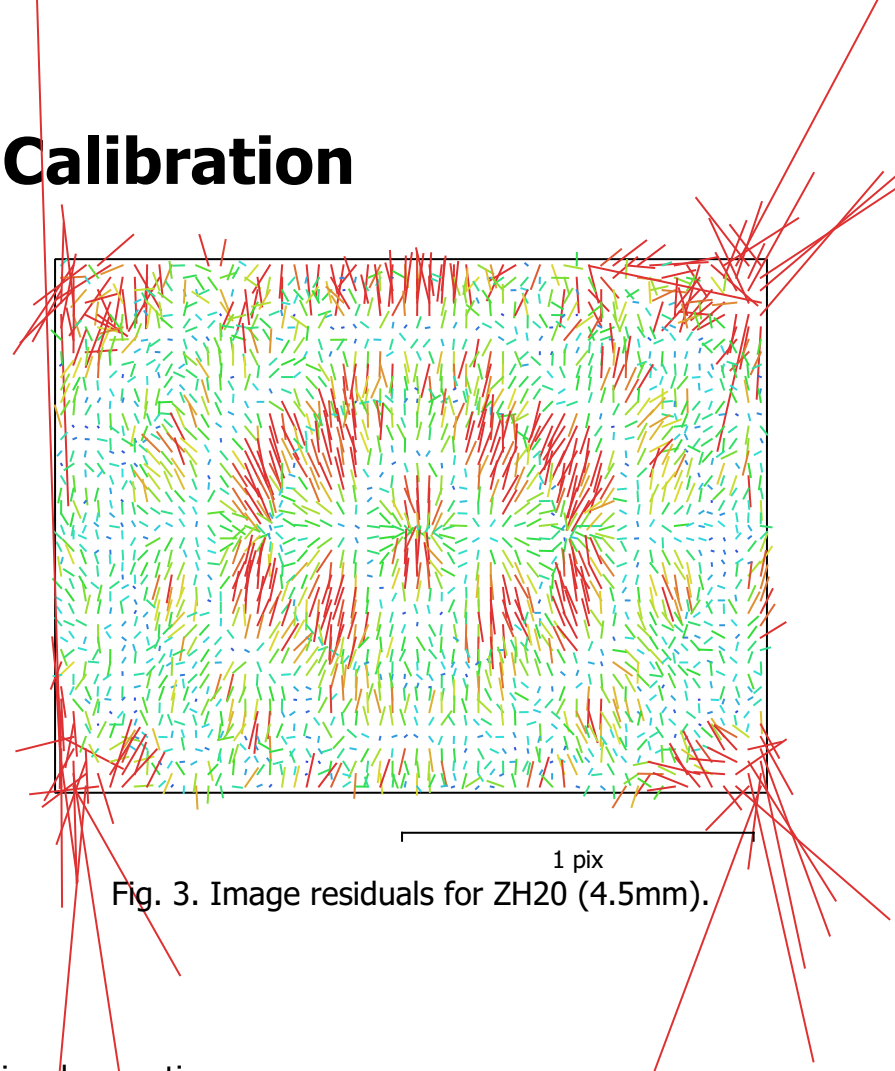


Fig. 3. Image residuals for ZH20 (4.5mm).

ZH20 (4.5mm)

328 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**4056 x 3040**	**4.5 mm**	**1.6 x 1.6 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**2975.72**	0.69	1.00	-0.07	0.04	-0.99	0.98	-0.95	0.92	-0.08	0.02
Cx	**37.5423**	0.11		1.00	0.02	0.07	-0.07	0.07	-0.07	0.92	0.04
Cy	**22.0068**	0.089			1.00	-0.03	0.03	-0.03	0.03	0.01	0.63
K1	**0.057096**	0.0022				1.00	-0.99	0.98	-0.95	0.09	-0.03
K2	**-0.0845759**	0.0073					1.00	-0.99	0.97	-0.09	0.03
K3	**-0.36137**	0.011						1.00	-0.99	0.09	-0.03
K4	**0.409085**	0.0061							1.00	-0.09	0.03
P1	**0.000265626**	1.3e-05								1.00	0.03
P2	**0.000154783**	8.9e-06									1.00

Table 3. Calibration coefficients and correlation matrix.

Camera Calibration

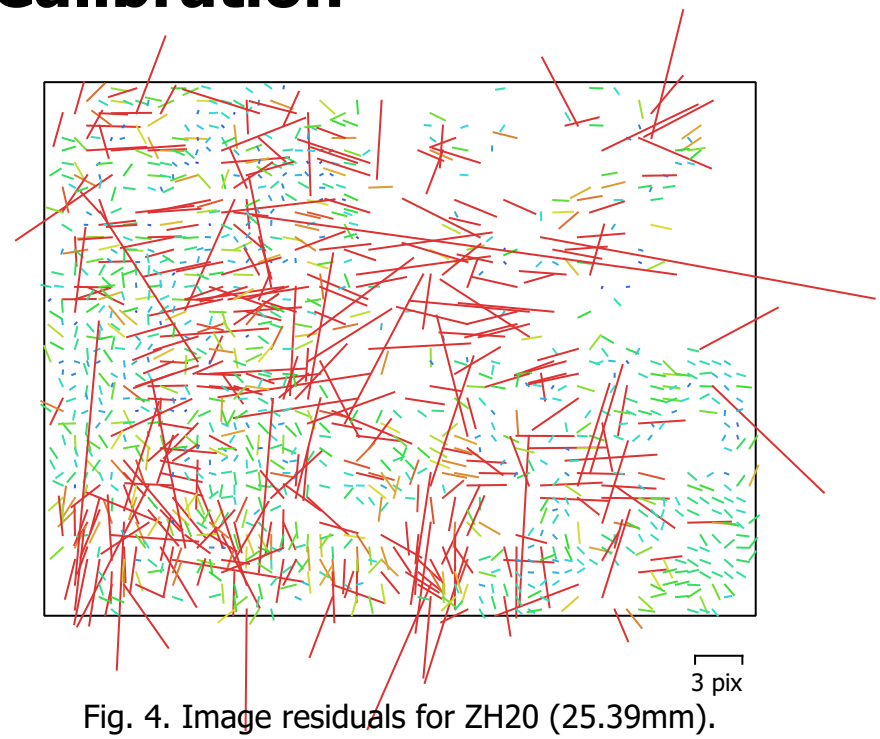


Fig. 4. Image residuals for ZH20 (25.39mm).

ZH20 (25.39mm)

9 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**5184 x 3888**	**25.39 mm**	**1.45 x 1.45 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**8246.01**	2e+02	1.00	0.37	0.07	-0.54	0.64	-0.68	0.69	0.21	0.14
Cx	**72.0655**	60		1.00	0.09	-0.72	0.74	-0.75	0.73	0.79	-0.00
Cy	**311.786**	26			1.00	-0.08	0.05	-0.02	-0.01	-0.01	0.15
K1	**-0.668631**	0.45				1.00	-0.98	0.95	-0.93	-0.61	0.11
K2	**12.9607**	6.8					1.00	-0.99	0.98	0.64	-0.09
K3	**-81.0412**	43						1.00	-1.00	-0.65	0.09
K4	**151.475**	93							1.00	0.65	-0.10
P1	**-0.00330712**	0.0043								1.00	0.08
P2	**0.0101221**	0.0025									1.00

Table 4. Calibration coefficients and correlation matrix.

Camera Locations

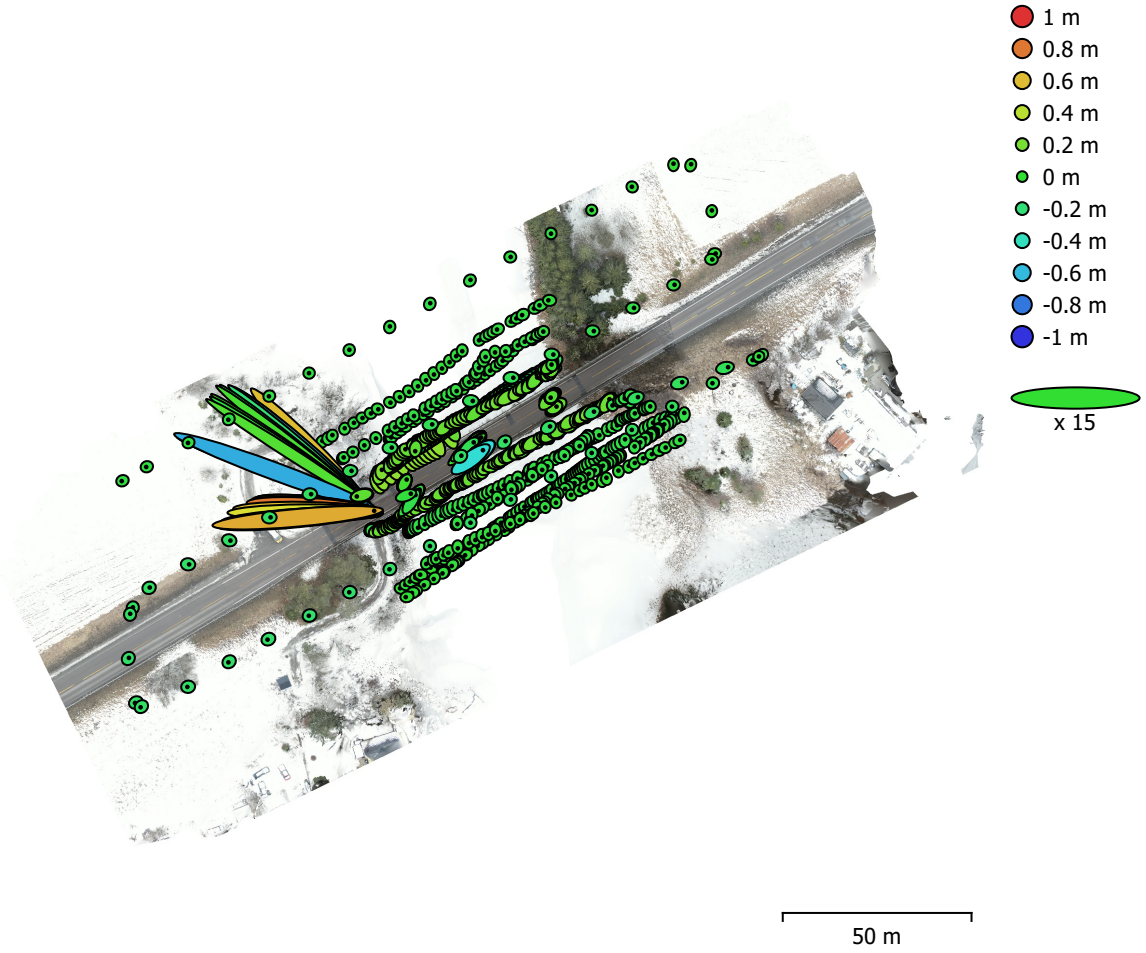


Fig. 5. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
39.1854	20.8749	14.2207	44.3989	46.6207

Table 5. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Ground Control Points

Fig. 6. GCP locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated GCP locations are marked with a dot or crossing.

Count	X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total (cm)
4	4.59528	1.84752	2.63531	4.95277	5.61024

Table 6. Control points RMSE.
X - Longitude, Y - Latitude, Z - Altitude.

Label	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Image (pix)
GCP1	-6.55857	-1.26271	-0.917412	6.74173	1.016 (15)
GCP2	0.1729	1.82001	-0.766921	1.98255	1.541 (20)
GCP3	-0.832326	2.86288	-5.08595	5.8954	2.559 (17)
GCP5	-6.38193	-0.741835	-0.694809	6.46236	0.659 (16)
Total	**4.59528**	**1.84752**	**2.63531**	**5.61024**	**1.633**

Table 7. Control points.

X - Longitude, Y - Latitude, Z - Altitude.

Digital Elevation Model

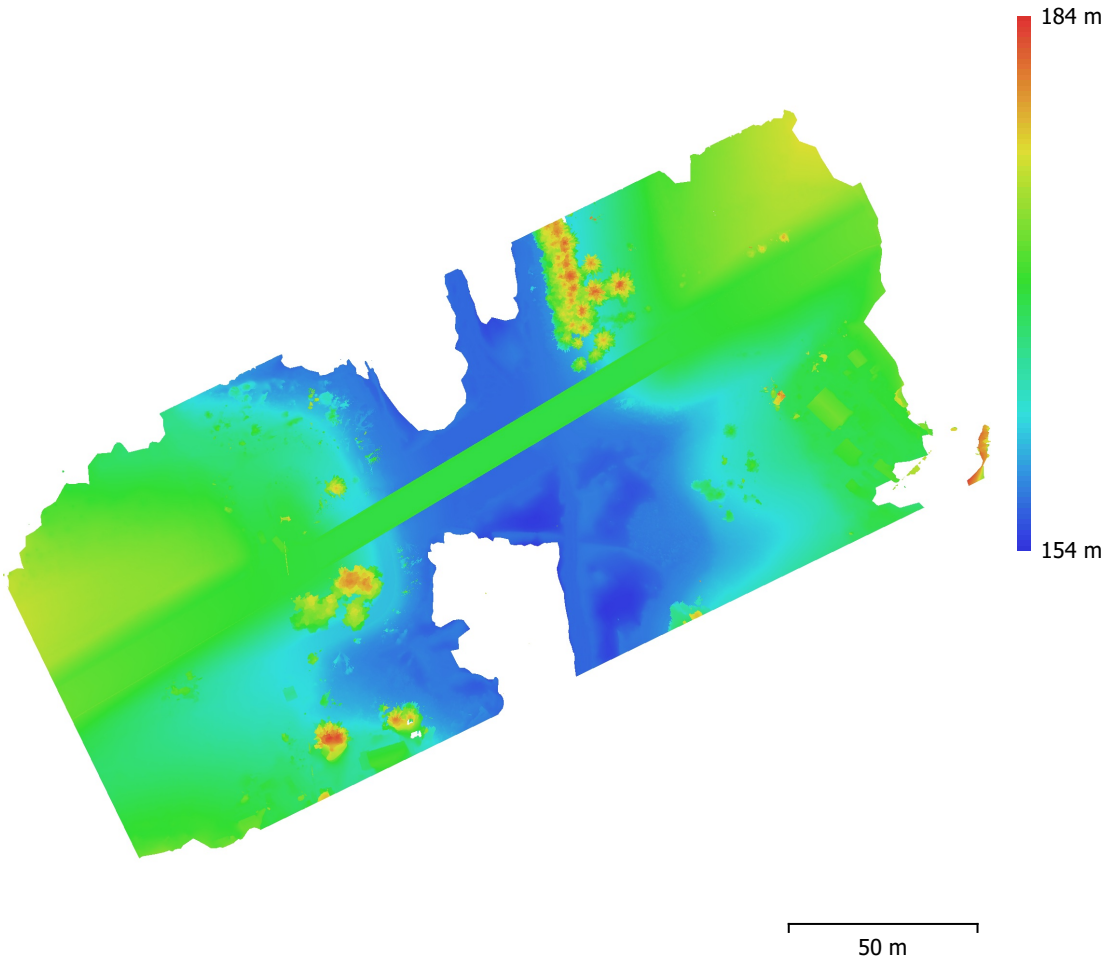


Fig. 7. Reconstructed digital elevation model.

Resolution:	1.16 cm/pix
Point density:	0.738 points/cm²

Processing Parameters

General	
Cameras	984
Aligned cameras	984
Markers	4
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	1,310,307 of 2,254,858
RMS reprojection error	0.136632 (0.768517 pix)
Max reprojection error	2.75629 (49.9651 pix)
Mean key point size	4.17721 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	3.9098
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Source
Key point limit	40,000
Key point limit per Mpx	1,000
Tie point limit	10,000
Exclude stationary tie points	No
Guided image matching	No
Adaptive camera model fitting	No
Matching time	25 minutes 37 seconds
Matching memory usage	3.39 GB
Alignment time	40 minutes 39 seconds
Alignment memory usage	985.19 MB
Optimization parameters	
Parameters	f, cx, cy, k1-k4, p1, p2
Fit additional corrections	Yes
Adaptive camera model fitting	No
Optimization time	20 minutes 57 seconds
Date created	2023:01:19 23:25:42
Software version	2.0.1.15925
File size	171.09 MB
Depth Maps	
Count	975
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 36 minutes
Memory usage	13.92 GB
Date created	2023:01:21 21:42:46
Software version	2.0.1.15925
File size	7.34 GB
Point Cloud	
Points	488,566,471
Point attributes	

Position	
Color	3 bands, uint8
Normal	
Confidence	
Point classes	
Created (never classified)	488,566,471
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 36 minutes
Memory usage	13.92 GB
Point cloud generation parameters	
Processing time	7 hours 43 minutes
Memory usage	24.80 GB
Date created	2023:01:22 05:26:05
Software version	2.0.1.15925
File size	6.93 GB
Model	
Faces	20,968,296
Vertices	10,512,530
Vertex colors	3 bands, uint8
Texture	8,192 x 8,192, 4 bands, uint8
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 36 minutes
Memory usage	13.92 GB
Reconstruction parameters	
Surface type	Arbitrary
Source data	Depth maps
Interpolation	Enabled
Strict volumetric masks	No
Processing time	1 hours 29 minutes
Memory usage	12.93 GB
Texturing parameters	
Mapping mode	Generic
Blending mode	Mosaic
Texture size	8,192
Enable hole filling	Yes
Enable ghosting filter	Yes
UV mapping time	6 minutes 18 seconds
UV mapping memory usage	3.98 GB
Blending time	5 hours 10 minutes
Blending memory usage	9.92 GB
Date created	2023:01:22 15:54:44
Software version	2.0.1.15925
File size	980.11 MB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.1 build 15925
OS	Windows 64 bit
RAM	31.76 GB
CPU	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
GPU(s)	Intel(R) UHD Graphics

Tillegg E

Agisoft Metasape Modell 3 rapport

Survey Data

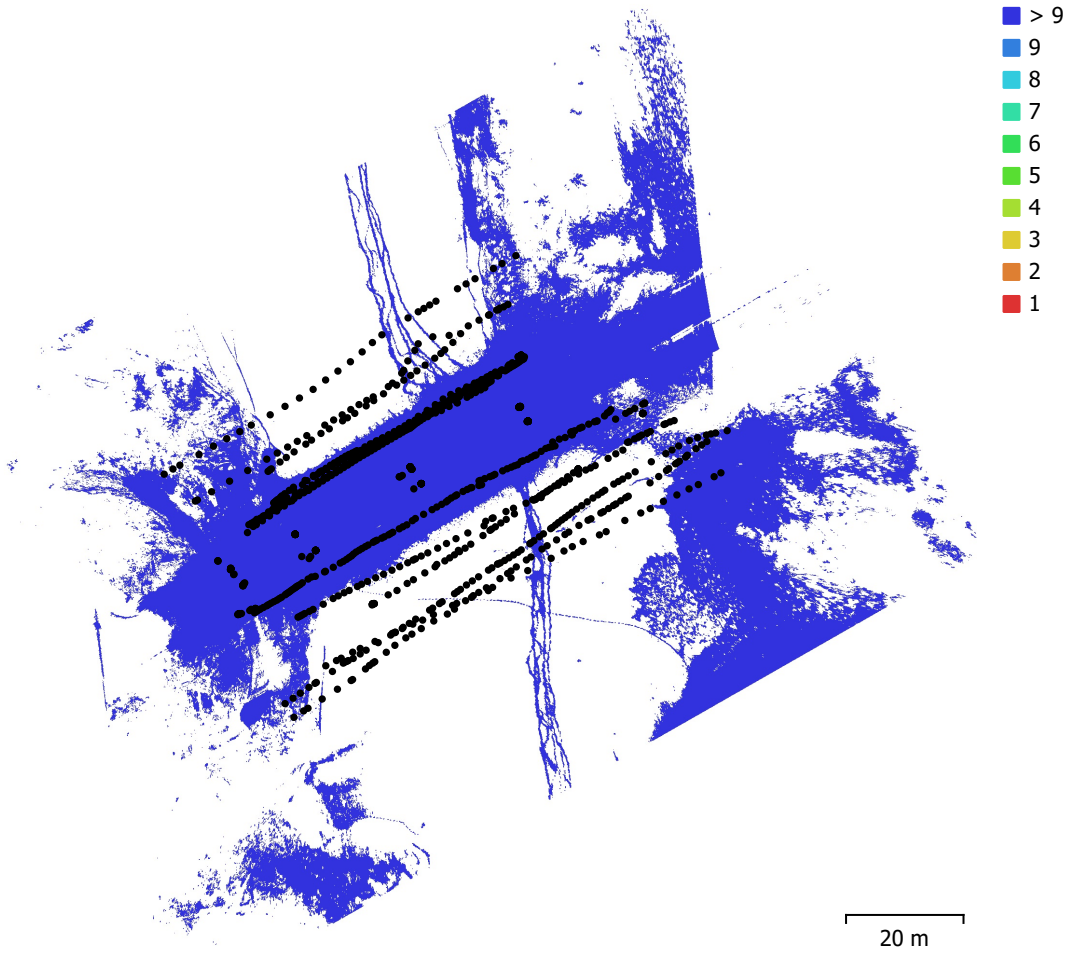


Fig. 1. Camera locations and image overlap.

Number of images:	921	Camera stations:	919
Flying altitude:	11.3 m	Tie points:	1,163,710
Ground resolution:	2.82 mm/pix	Projections:	4,740,587
Coverage area:	5.18e+03 m²	Reprojection error:	1.74 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
ZenmuseP1 (35mm)	8192 x 5460	35 mm	4.39 x 4.39 µm	No
ZH20 (4.5mm)	4056 x 3040	4.5 mm	1.6 x 1.6 µm	No
ZH20 (25.39mm)	5184 x 3888	25.39 mm	1.45 x 1.45 µm	No

Table 1. Cameras.

Camera Calibration

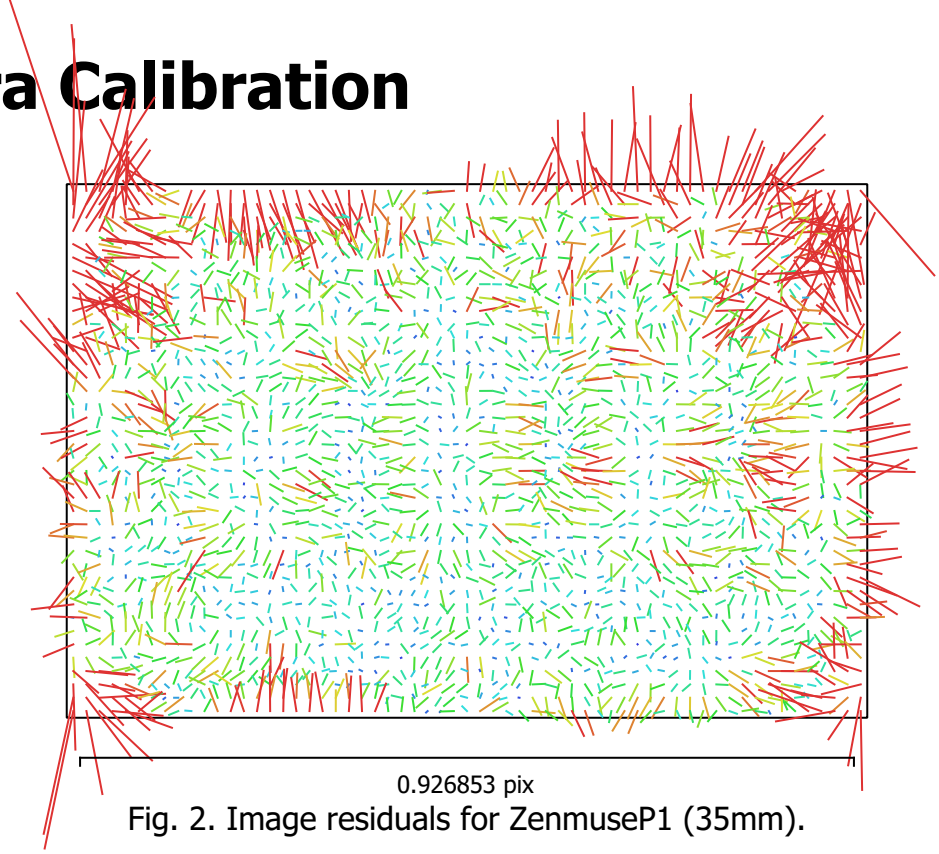


Fig. 2. Image residuals for ZenmuseP1 (35mm).

ZenmuseP1 (35mm)

584 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**8192 x 5460**	**35 mm**	**4.39 x 4.39 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**8191.54**	0.26	1.00	0.04	0.04	-0.98	0.96	-0.93	0.89	0.03	0.03
Cx	**-9.86552**	0.13		1.00	0.01	-0.03	0.02	-0.02	0.02	0.94	0.00
Cy	**7.77571**	0.11			1.00	-0.04	0.04	-0.04	0.04	0.01	0.93
K1	**-0.041486**	0.00057				1.00	-0.99	0.97	-0.94	-0.03	-0.03
K2	**0.117491**	0.0037					1.00	-0.99	0.97	0.02	0.04
K3	**-0.410658**	0.01						1.00	-0.99	-0.02	-0.04
K4	**0.374342**	0.011							1.00	0.02	0.04
P1	**-0.000368528**	6.2e-06								1.00	0.00
P2	**0.00164675**	5.6e-06									1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Calibration

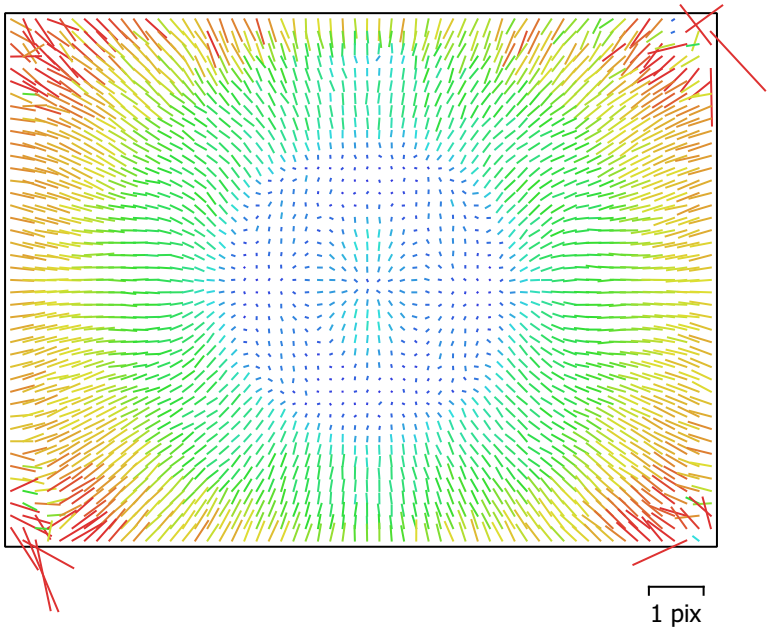


Fig. 3. Image residuals for ZH20 (4.5mm).

ZH20 (4.5mm)

328 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**4056 x 3040**	**4.5 mm**	**1.6 x 1.6 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**2994.19**	0.85	1.00	-0.07	0.05	-0.99	0.97	-0.94	0.90	-0.07	0.02
Cx	**35.7717**	0.14		1.00	0.03	0.07	-0.08	0.08	-0.08	0.93	0.04
Cy	**23.835**	0.11			1.00	-0.03	0.03	-0.03	0.03	0.02	0.59
K1	**0.0156761**	0.0027				1.00	-0.99	0.97	-0.94	0.08	-0.02
K2	**0.0833773**	0.0093					1.00	-0.99	0.97	-0.08	0.02
K3	**-0.677636**	0.014						1.00	-0.99	0.09	-0.02
K4	**0.635425**	0.0084							1.00	-0.08	0.02
P1	**3.43608e-05**	1.6e-05								1.00	0.03
P2	**0.000150617**	1.1e-05									1.00

Table 3. Calibration coefficients and correlation matrix.

Camera Calibration

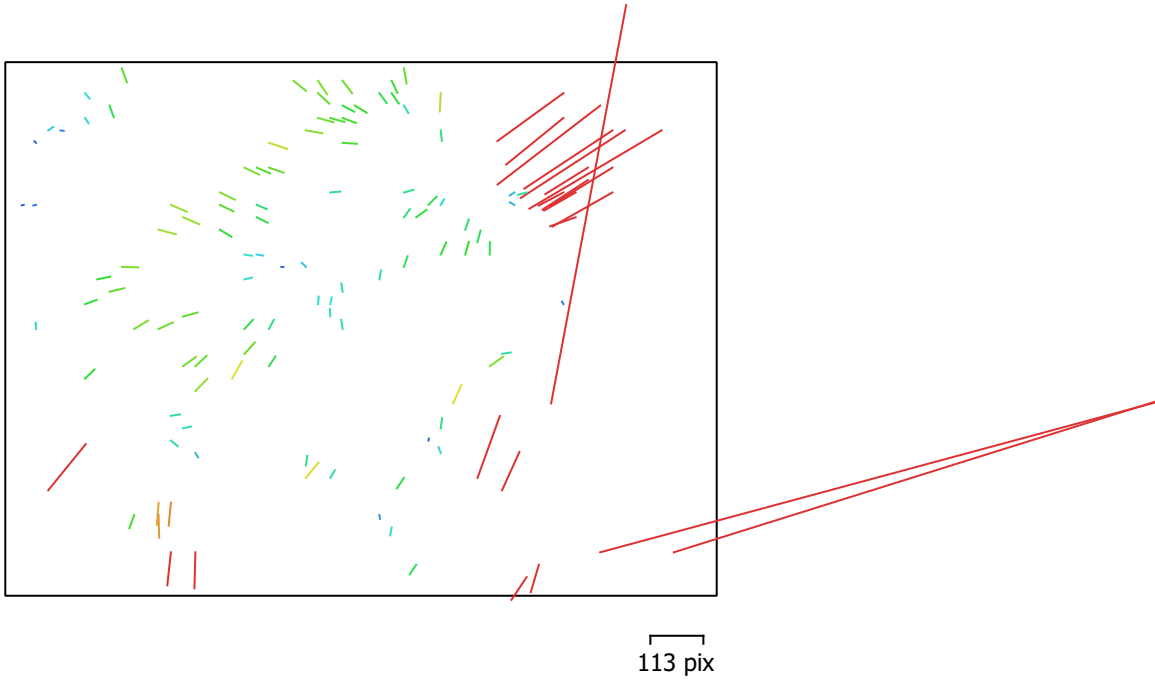


Fig. 4. Image residuals for ZH20 (25.39mm).

ZH20 (25.39mm)

9 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**5184 x 3888**	**25.39 mm**	**1.45 x 1.45 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**15780.1**	1.7e+03	1.00	-0.72	0.20	-0.97	0.96	-0.93	0.92	0.40	0.30
Cx	**-2129.93**	7.9e+02		1.00	-0.41	0.70	-0.83	0.85	-0.82	-0.01	0.11
Cy	**-457.457**	5e+02			1.00	-0.10	0.21	-0.23	0.22	-0.43	-0.25
K1	**21.4335**	20				1.00	-0.96	0.94	-0.93	-0.54	-0.45
K2	**-521.594**	8.2e+02					1.00	-1.00	0.99	0.31	0.22
K3	**6103.04**	1.4e+04						1.00	-1.00	-0.27	-0.16
K4	**-22648.4**	8.1e+04							1.00	0.28	0.17
P1	**-0.777145**	0.33								1.00	0.87
P2	**-0.0326105**	0.13									1.00

Table 4. Calibration coefficients and correlation matrix.

Camera Locations

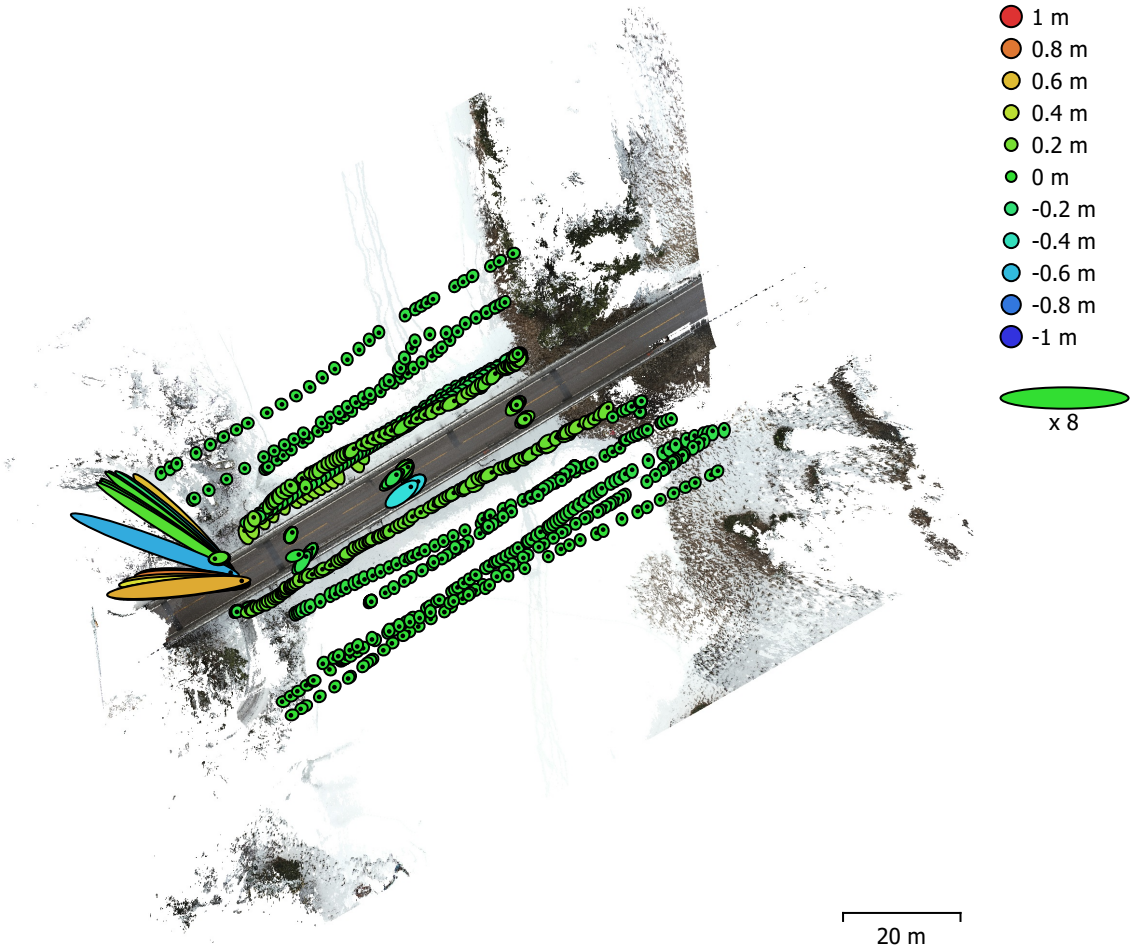


Fig. 5. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
39.9615	21.0937	14.1443	45.187	47.349

Table 5. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Ground Control Points

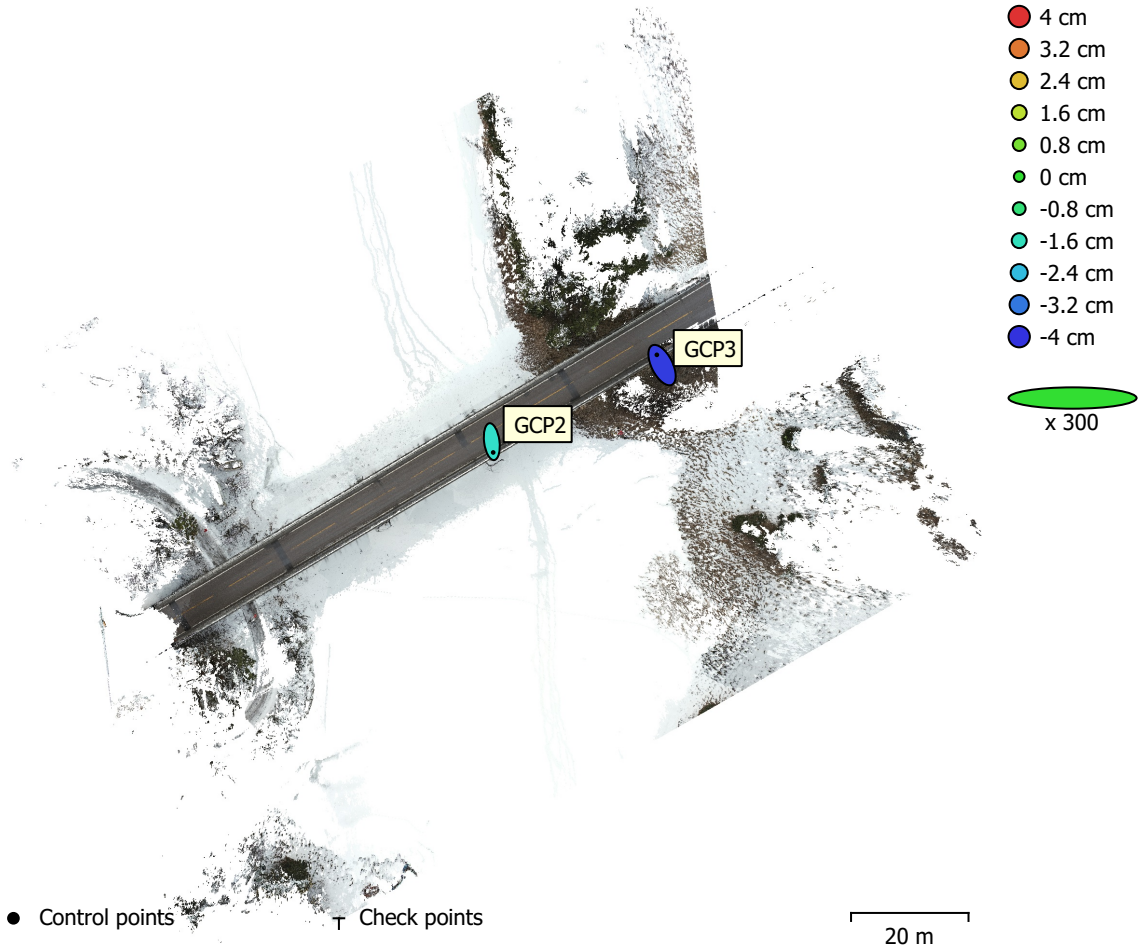


Fig. 6. GCP locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated GCP locations are marked with a dot or crossing.

Count	X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total (cm)
2	0.430591	1.19632	3.01709	1.27145	3.27405

Table 6. Control points RMSE.
X - Longitude, Y - Latitude, Z - Altitude.

Label	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Image (pix)
GCP2	0.158303	-1.22755	-1.75453	2.14716	1.905 (40)
GCP3	-0.588011	1.16424	-3.88937	4.10225	0.863 (11)
Total	**0.430591**	**1.19632**	**3.01709**	**3.27405**	**1.734**

Table 7. Control points.
X - Longitude, Y - Latitude, Z - Altitude.

Digital Elevation Model

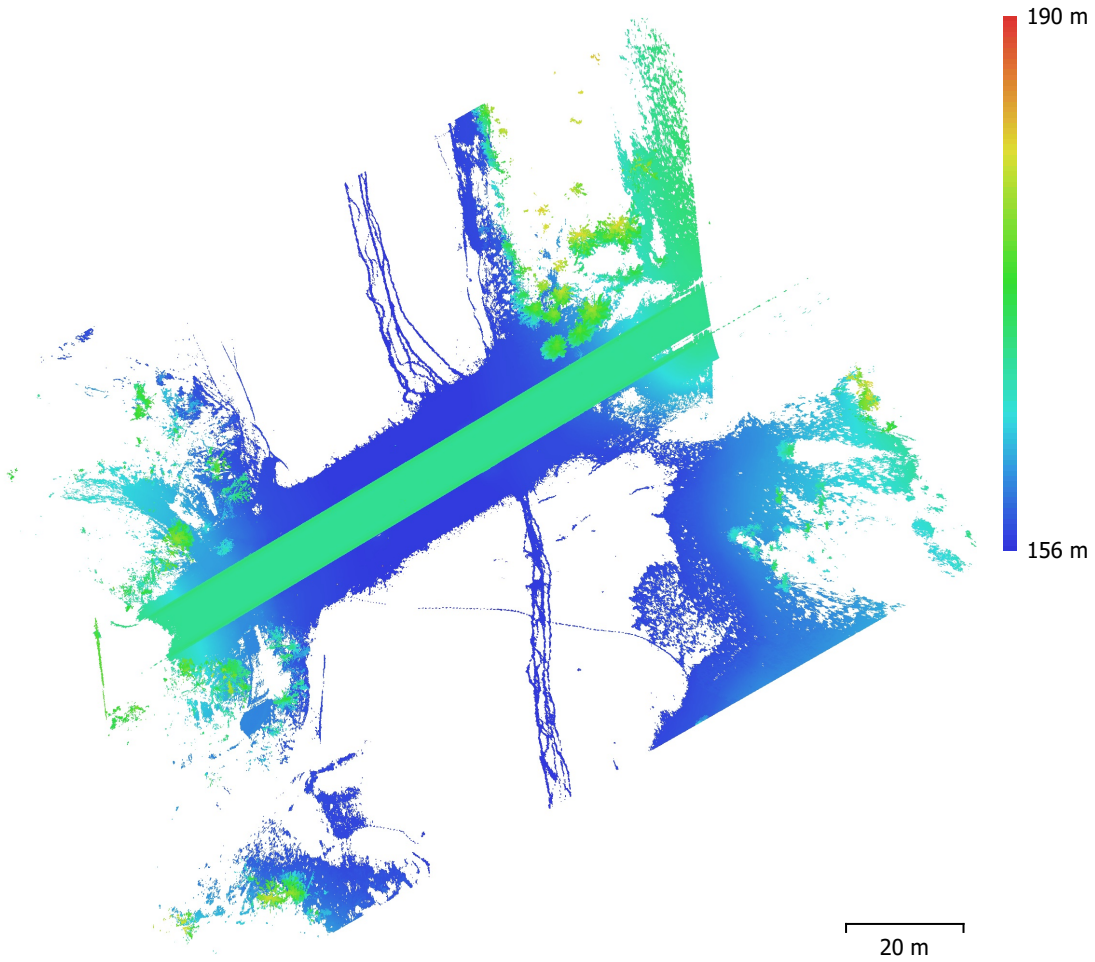


Fig. 7. Reconstructed digital elevation model.

Resolution:	unknown
Point density:	unknown

Processing Parameters

General	
Cameras	921
Aligned cameras	919
Markers	2
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	1,163,710 of 2,096,955
RMS reprojection error	0.244055 (1.73516 pix)
Max reprojection error	318.408 (2493.84 pix)
Mean key point size	4.1563 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	3.91737
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Source
Key point limit	40,000
Key point limit per Mpx	1,000
Tie point limit	10,000
Exclude stationary tie points	No
Guided image matching	No
Adaptive camera model fitting	No
Matching time	24 minutes 10 seconds
Matching memory usage	3.15 GB
Alignment time	32 minutes 29 seconds
Alignment memory usage	909.87 MB
Optimization parameters	
Parameters	f, cx, cy, k1-k4, p1, p2
Fit additional corrections	Yes
Adaptive camera model fitting	No
Optimization time	22 minutes 36 seconds
Date created	2023:01:20 13:29:28
Software version	2.0.1.15925
File size	158.10 MB
Depth Maps	
Count	913
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	4 hours 19 minutes
Memory usage	13.13 GB
Date created	2023:01:20 23:10:08
Software version	2.0.1.15925
File size	6.83 GB
Point Cloud	
Points	350,031,037
Point attributes	

Position	
Color	3 bands, uint8
Normal	
Confidence	
Point classes	
Created (never classified)	350,031,037
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	4 hours 19 minutes
Memory usage	13.13 GB
Point cloud generation parameters	
Processing time	6 hours 48 minutes
Memory usage	26.75 GB
Date created	2023:01:21 05:58:42
Software version	2.0.1.15925
File size	5.02 GB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.1 build 15925
OS	Windows 64 bit
RAM	31.76 GB
CPU	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
GPU(s)	Intel(R) UHD Graphics Quadro T1000

Tillegg F

Agisoft Metasape Modell 4 rapport

Survey Data

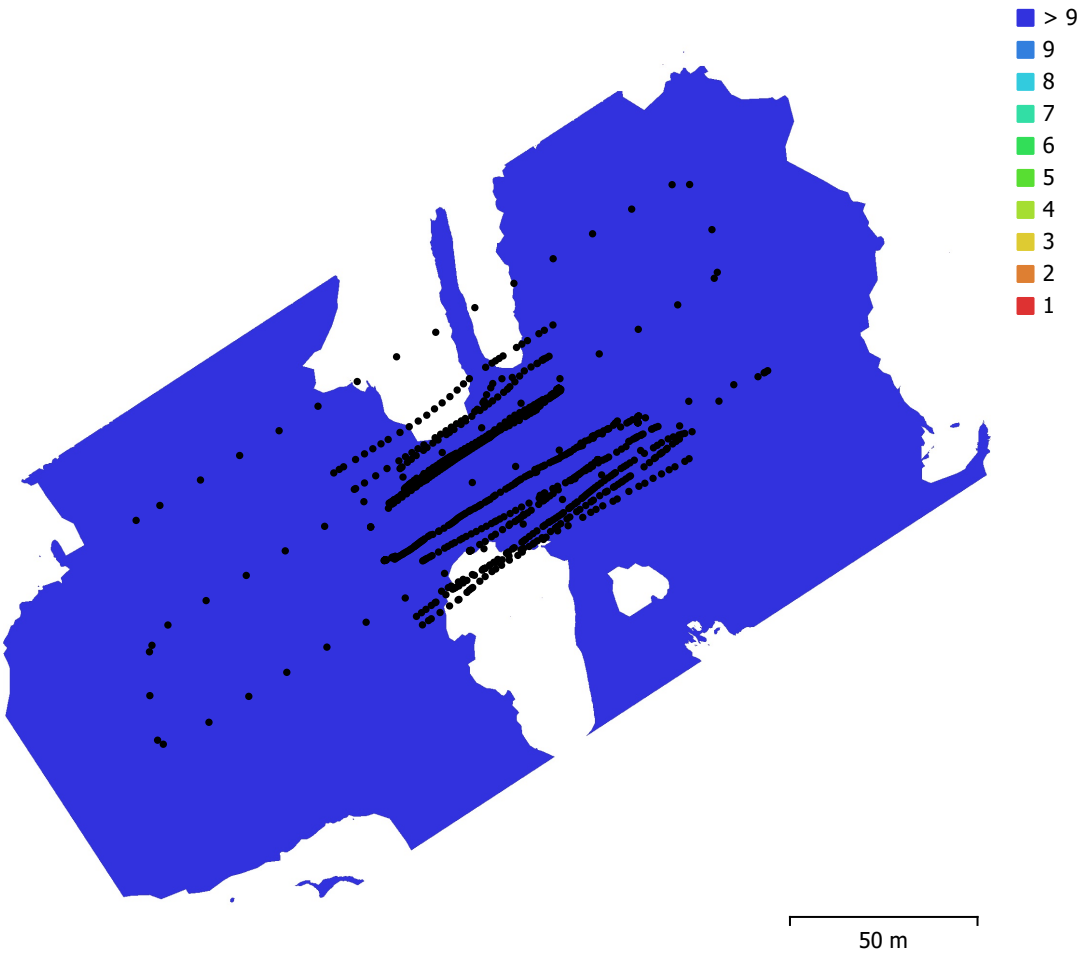


Fig. 1. Camera locations and image overlap.

Number of images:	845	Camera stations:	844
Flying altitude:	14.6 m	Tie points:	1,247,026
Ground resolution:	3.43 mm/pix	Projections:	5,136,174
Coverage area:	0.0288 km²	Reprojection error:	0.732 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
ZH20 (4.5mm)	4056 x 3040	4.5 mm	1.6 x 1.6 µm	No
ZH20 (25.39mm)	5184 x 3888	25.39 mm	1.45 x 1.45 µm	No
ZenmuseP1 (35mm)	8192 x 5460	35 mm	4.39 x 4.39 µm	No

Table 1. Cameras.

Camera Calibration

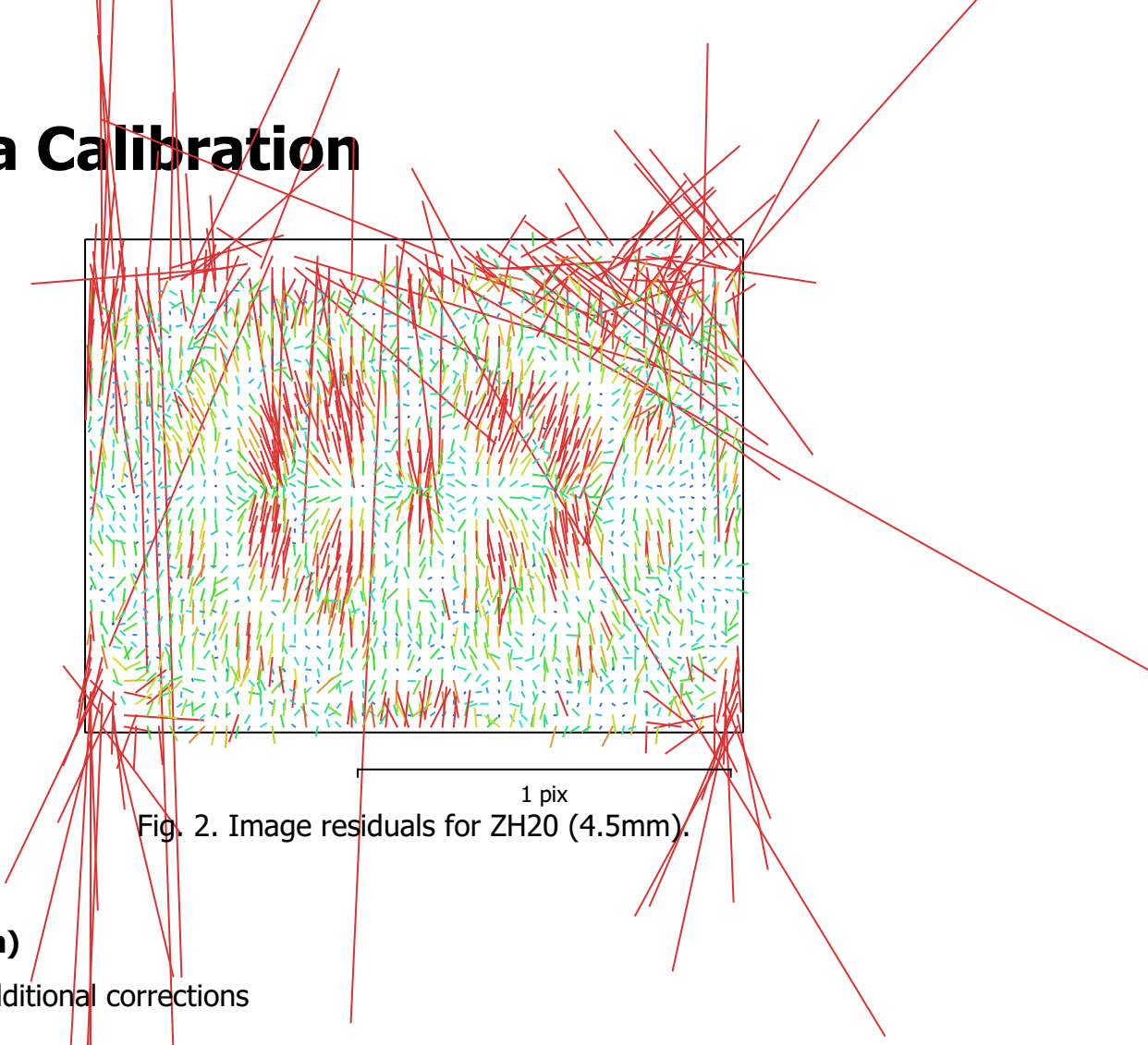


Fig. 2. Image residuals for ZH20 (4.5mm).

ZH20 (4.5mm)

189 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**4056 x 3040**	**4.5 mm**	**1.6 x 1.6 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**2984.36**	1.2	1.00	-0.10	0.08	-0.99	0.98	-0.96	0.92	-0.10	-0.03
Cx	**33.056**	0.12		1.00	0.04	0.11	-0.12	0.12	-0.12	0.89	0.01
Cy	**31.8878**	0.11			1.00	-0.07	0.07	-0.07	0.08	0.02	0.23
K1	**0.0489052**	0.0035				1.00	-1.00	0.98	-0.96	0.11	0.03
K2	**-0.00431155**	0.011					1.00	-1.00	0.98	-0.12	-0.03
K3	**-0.556477**	0.016						1.00	-0.99	0.12	0.02
K4	**0.559709**	0.0086							1.00	-0.13	-0.02
P1	**0.000144892**	1.3e-05								1.00	0.01
P2	**-2.44903e-05**	7.9e-06									1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Calibration

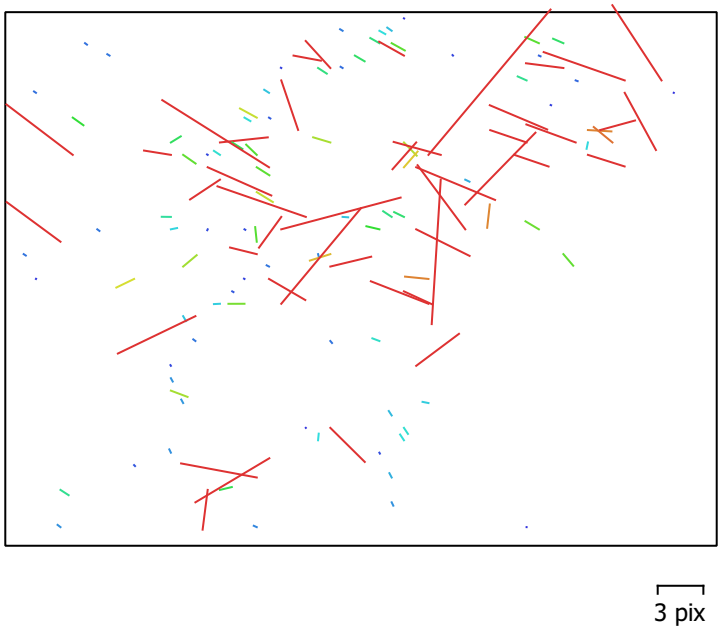


Fig. 3. Image residuals for ZH20 (25.39mm).

ZH20 (25.39mm)

9 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**5184 x 3888**	**25.39 mm**	**1.45 x 1.45 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**12933.8**	4.9e+02	1.00	-0.25	0.24	-0.39	-0.25	0.40	-0.44	-0.41	0.22
Cx	**-545.808**	81		1.00	-0.07	0.03	0.15	-0.18	0.19	-0.28	0.10
Cy	**-52.1868**	1e+02			1.00	0.18	-0.39	0.43	-0.45	-0.45	-0.24
K1	**46.7891**	9.1				1.00	-0.78	0.65	-0.60	-0.13	-0.12
K2	**-622.67**	1.4e+03					1.00	-0.98	0.96	0.44	-0.04
K3	**-71155.3**	8e+04						1.00	-1.00	-0.51	0.11
K4	**1.70473e+06**	1.3e+06							1.00	0.52	-0.13
P1	**0.407279**	0.088								1.00	-0.46
P2	**-0.325646**	0.069									1.00

Table 3. Calibration coefficients and correlation matrix.

Camera Calibration

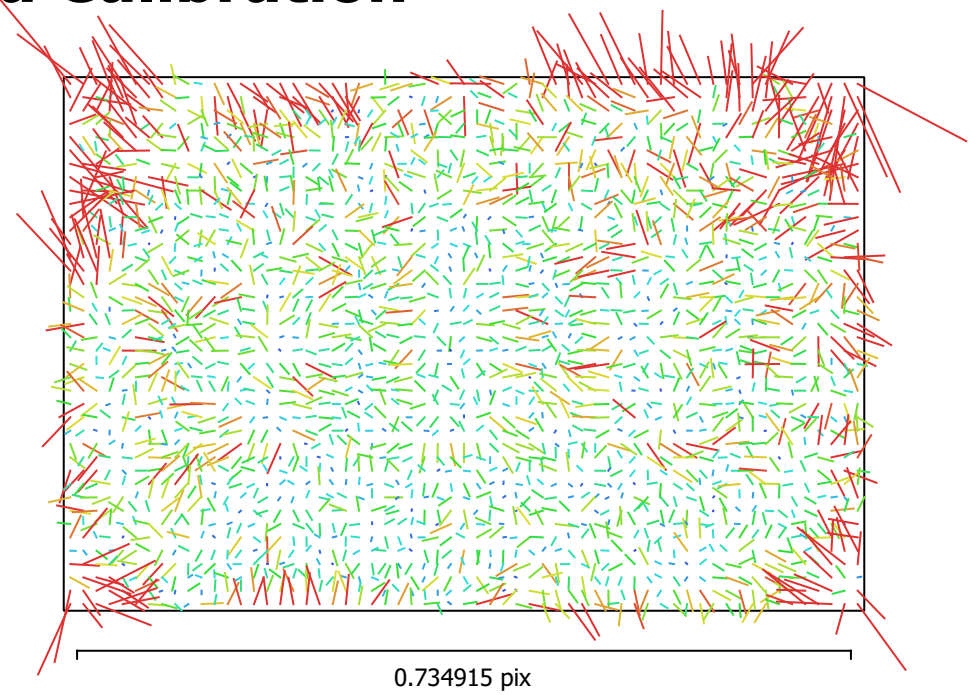


Fig. 4. Image residuals for ZenmuseP1 (35mm).

ZenmuseP1 (35mm)

647 images, additional corrections

Type	Resolution	Focal Length	Pixel Size
Frame	**8192 x 5460**	**35 mm**	**4.39 x 4.39 µm**

	Value	Error	F	Cx	Cy	K1	K2	K3	K4	P1	P2
F	**8190.49**	0.17	1.00	0.03	0.03	-0.98	0.96	-0.93	0.89	0.02	0.02
Cx	**-8.34913**	0.082		1.00	0.00	-0.02	0.02	-0.02	0.01	0.96	-0.00
Cy	**7.88193**	0.071			1.00	-0.03	0.03	-0.03	0.03	0.00	0.95
K1	**-0.0399727**	0.00038				1.00	-0.99	0.97	-0.94	-0.02	-0.03
K2	**0.104273**	0.0025					1.00	-0.99	0.97	0.02	0.03
K3	**-0.365699**	0.0069						1.00	-0.99	-0.01	-0.03
K4	**0.312472**	0.0072							1.00	0.01	0.03
P1	**-0.00039643**	4.2e-06								1.00	-0.00
P2	**0.0017386**	3.9e-06									1.00

Table 4. Calibration coefficients and correlation matrix.

Camera Locations

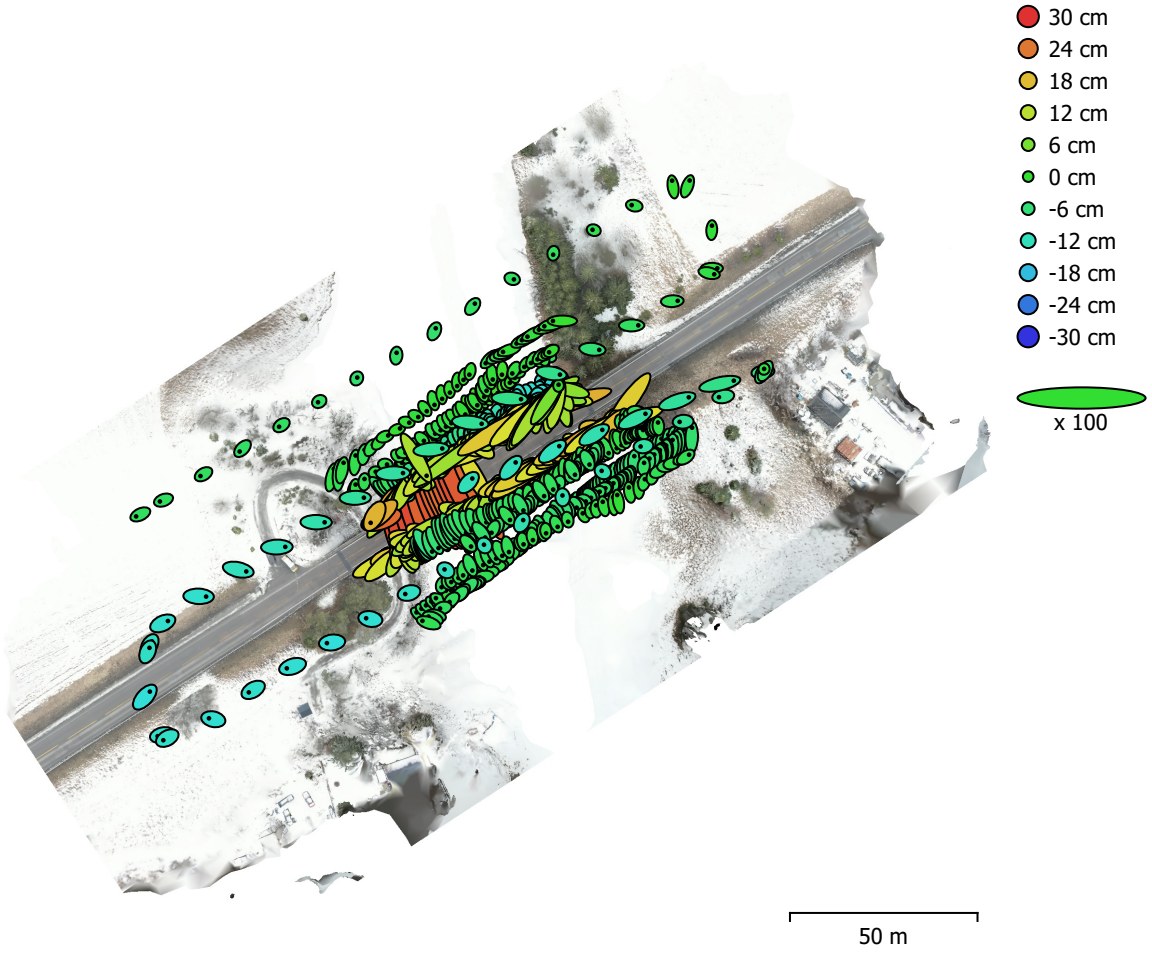


Fig. 5. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
3.73107	3.80374	9.17028	5.32816	10.6058

Table 5. Average camera location error.
X - Easting, Y - Northing, Z - Altitude.

Ground Control Points

Fig. 6. GCP locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated GCP locations are marked with a dot or crossing.

Count	X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total (cm)
4	3.8594	1.2727	3.34285	4.06383	5.26207

Table 6. Control points RMSE.
X - Easting, Y - Northing, Z - Altitude.

Label	X error (cm)	Y error (cm)	Z error (cm)	Total (cm)	Image (pix)
GCP1	-5.48579	-2.05599	2.73202	6.46413	0.929 (15)
GCP2	0.858244	0.29468	4.85018	4.93434	1.809 (13)
GCP3	-1.23806	0.477396	-1.13049	1.74319	1.723 (14)
GCP5	-5.21696	-1.39185	3.52594	6.44873	0.531 (13)
Total	**3.8594**	**1.2727**	**3.34285**	**5.26207**	**1.353**

Table 7. Control points.

X - Easting, Y - Northing, Z - Altitude.

Digital Elevation Model

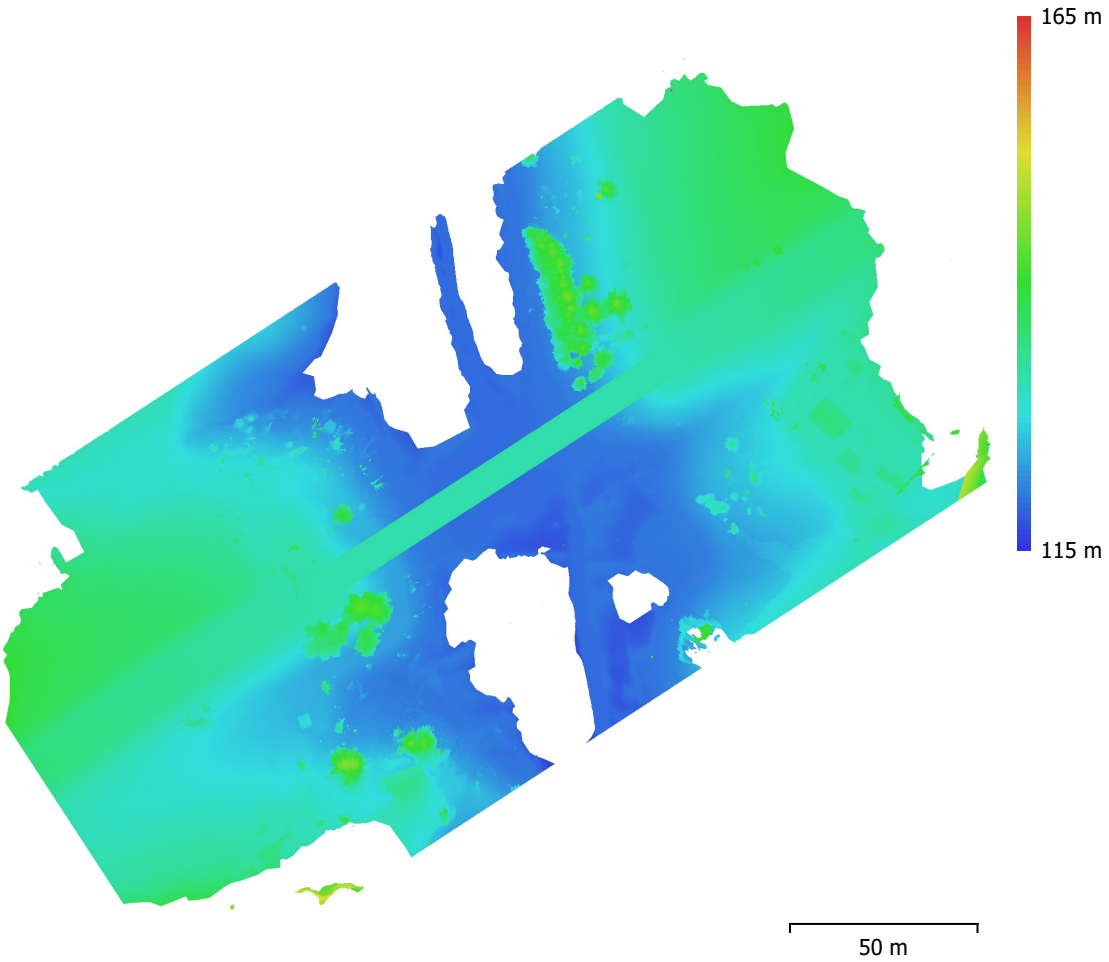


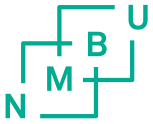
Fig. 7. Reconstructed digital elevation model.

Resolution:	1.15 cm/pix
Point density:	0.757 points/cm²

Processing Parameters

General	
Cameras	845
Aligned cameras	844
Markers	4
Coordinate system	ETRS89 / UTM zone 32N + NN2000 height (EPSG::5972)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	1,247,026 of 1,896,146
RMS reprojection error	0.130603 (0.731768 pix)
Max reprojection error	2.52591 (46.7237 pix)
Mean key point size	4.13757 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	4.13467
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Source
Key point limit	45,000
Key point limit per Mpx	1,000
Tie point limit	10,000
Exclude stationary tie points	No
Guided image matching	No
Adaptive camera model fitting	No
Matching time	28 minutes 6 seconds
Matching memory usage	4.24 GB
Alignment time	28 minutes 35 seconds
Alignment memory usage	674.46 MB
Optimization parameters	
Parameters	f, cx, cy, k1-k4, p1, p2
Fit additional corrections	Yes
Adaptive camera model fitting	No
Optimization time	8 minutes 24 seconds
Date created	2023:01:18 19:25:40
Software version	2.0.1.15925
File size	153.71 MB
Depth Maps	
Count	837
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 52 minutes
Memory usage	11.11 GB
Date created	2023:01:18 03:55:41
Software version	2.0.1.15925
File size	6.99 GB
Point Cloud	
Points	132,420,230
Point attributes	

Position	
Color	3 bands, uint8
Normal	
Confidence	
Point classes	
Created (never classified)	132,420,230
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 22 minutes
Memory usage	11.93 GB
Point cloud generation parameters	
Processing time	7 hours 27 minutes
Memory usage	25.16 GB
Date created	2023:01:20 02:08:32
Software version	2.0.1.15925
File size	7.48 GB
Model	
Faces	20,716,906
Vertices	10,406,268
Vertex colors	3 bands, uint8
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Max neighbors	16
Processing time	3 hours 52 minutes
Memory usage	11.11 GB
Reconstruction parameters	
Surface type	Arbitrary
Source data	Depth maps
Interpolation	Enabled
Strict volumetric masks	No
Processing time	1 hours 23 minutes
Memory usage	12.29 GB
Date created	2023:01:18 05:17:59
Software version	2.0.1.15925
File size	878.03 MB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.1 build 15925
OS	Windows 64 bit
RAM	31.76 GB
CPU	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
GPU(s)	Intel(R) UHD Graphics Quadro T1000



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway