

## Rapport Lab 4 – stitching

### Inledning

Målet med denna laboration var att med valfrit tillvägagångssätt *stitcha* ihop 3 stycken bilder till en panoramabild. De tre bilder vi valde att använda syns nedan i *figur 1*. Rapporten kommer ta upp de tre olika sätt vi använde för att sätta ihop dessa bilder: *korrelation*, *Hough transform* & *SIFT*.



Figur 1: De tre bilder som skall sättas ihop

### Metod 1 – Korrelation

Liknande lab 2 då man använde korrelation för att lokalisera zlatans öga och mun i en bild på hela ansiktet. Ett försök med samma metod för att *stitcha* ihop de tre bilderna gjordes även i denna lab, då det ansågs svårt så började vi med att sätta ihop två av bilderna. Målet är att hitta tydliga unika *subbilder* i *bild 1* och sedan lokalisera samma/liknande sub-bilder i *bild 2*. Nedan i *figur 2* visas ett exempel på sub-bilder som valts ut.



Figur 2: Sub-bilder som jämförs i bilderna

Sub-bilderna jämförs med varandra och då man tyckt att man har hittat samma punkt i båda bilderna kan man ta ut en transformationsmatris. Matrisen i fråga kan man lätt få fram med hjälp av *backslash* funktionen i matlab. Då vi till slut visste att tre bilder skulle sättas ihop så valdes det att båda sidobilderna skall transformeras efter mittbilden, detta för att undvika att bilden skulle skjuvas för mycket.

Resultatet syns i *figur 3* nedan då bilden gått igenom en medianfiltrering, notera att bilden är i gråskala, detta gjordes för att färgkanalerna kan försummas, detta gäller för större delen av vårt arbete.. Det märktes snabbt att denna metod inte lämpade sig så väl för våra bilder. Om man tittar på *bild 1* och *bild 2* så ser man att det är väldigt svårt att plocka ut tydliga unika sub-bilder i den undre delen av bilderna, det betyder att de flesta referenspunkterna hamnar i övre delen vilket gjorde

Fredrik Johnson, Frejo989  
LaborationsPartner: Rasmus Haapaoja, Anton Albèrt

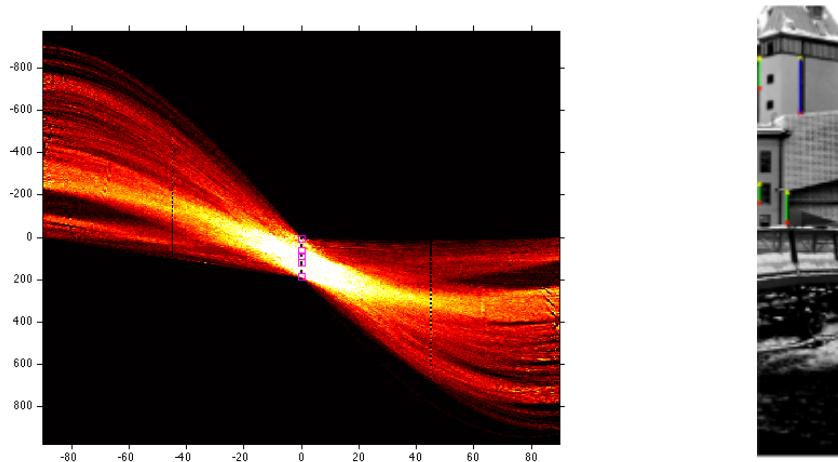
transformen väldigt dålig.



Figur 3: Resultat med korrelation

## Metod 2 – Houghtransform

Försökt nummer två gick ut på att använda *Hough Transform* vilket går ut på att istället för att lokalisera punkter i en bild så letar man efter linjer, cirklar eller andra parametriska kurvor. Nyckelorden här är parametriska kurvor, det vill säga att vi måste transformera vår bilds koordinater till ett annat koordinatsystem så de beror på parametriska värden. Då man sedan mäter upp bilden i dess nya koordinatsystem ser man flera sinuskurvor, där dessa sinuskurvor skär varandra existerar en linje i vår orginalbild. De magentafärgade markeringarna i *figur 4* representerar en linje i vår bild, de blå och grönmärkade linjerna i bilden till höger är de bilder som hittats.



Figur 4: Hough-transformen, visar där sinuskurvor skär med motstående linje i orginalbilden

Vi tyckte att denna metod verkade lovande då den enkelt kunde urskilja linjer i varje bild. Men dessa linjer var inte alltid desamma och när det väl var samma linje så var det istället olika långa så de kunde inte matchas ihop då heller. Efter många försök då vi verkligen trodde att det skulle gå och lösa med denna metod beslöt vi oss för att lämna den och leta efter något annat sätt att lösa problemet på.

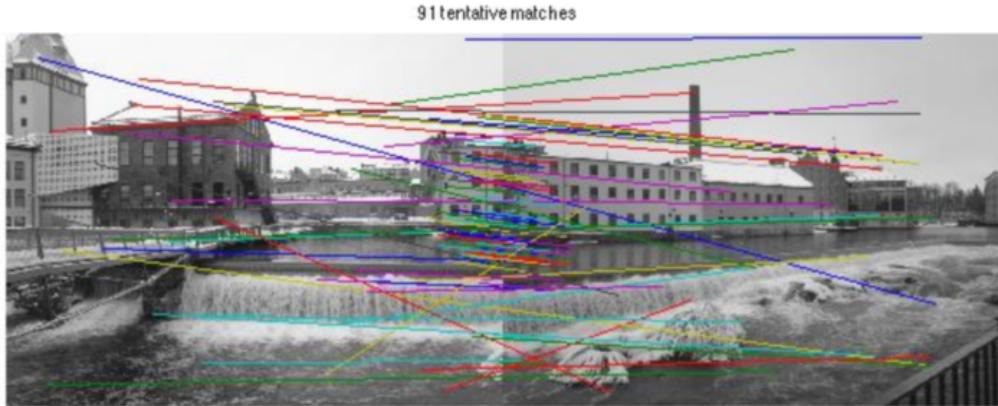
## Metod 3 – SIFT (fusk)

Efter att övergivit både korrelation och Houghtransformen så började vi leta efter andra sätt att lösa problemet på. Till slut hittades SIFT(Scale Invariant Feature Transform). Denna metod liknar det vi försökte med i början, korrelation, fast på ett betydligt mer avancerat tillvägagångssätt. Målet är att i två bilder hitta *keypoints*, men till skillnad från innan representeras dessa som en cirkelformad subbild. Denna cirkulära bild har attribut som vinkel och storlek(radien) som sparats för varje sub-bild, alla bilder matchas ihop utifrån de attribut som sparats ner. I *figur X* visas matchningarna efter att en

Fredrik Johnson, Frejo989

LaborationsPartner: Rasmus Haapaoja, Anton Albèrt

del av SIFT-algoritmen har körts.



Figur 5: SIFT-matchningar innan filtrering

Vissa av matchningarna verkar inte stämma helt korrekt, detta beror på att attributen som finns i varje delcirkel kan variera i bilderna. För att få en bättre matchning gör SIFT ytterligare jämförelser för att sälla bort punkter som inte har en tillräckligt bra matchning i vardera bild. När algoritmen är klar så finns de punkter som har tillräckligt bra matchning kvar, visas i *figur 6*,



Figur 6: SIFT-matchningar efter filtrering

Fredrik Johnson, Frejo989

LaborationsPartner: Rasmus Haapaoja, Anton Albèrt

Då bilderna sätts ihop skapas det en helt ny bild där värdena för den nya bilden kopieras in ifrån rätt bild. Resultatet av SIFT syns nedan i *figur 7*.



*Figur 7: Resultat med SIFT*

Då SIFT användes kom vår grupp fram till ett resultat som vi var nöjda med, men om man kollar noga där två bilder möts så ser man skarvar. Detta borde gå att lösa genom att låta ett filter löpa över bilden vilket bara ser till att ta värde från en av bilderna. Det skall poängteras att gruppen skrev inte SIFT-algoritmen själv utan använde ett färdigt bibliotek som modifierades efter behov.