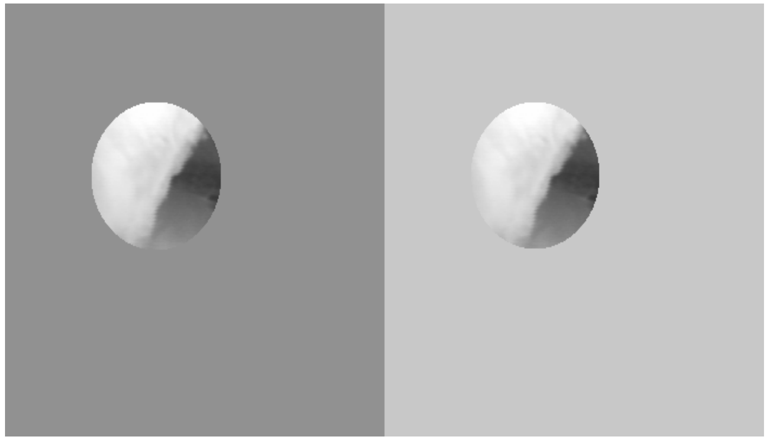
Bildverarbeitung und Algorithmen

Prof. Dr. Wolfgang Konen

Image Mosaicing & Motion Estimation



Überblick

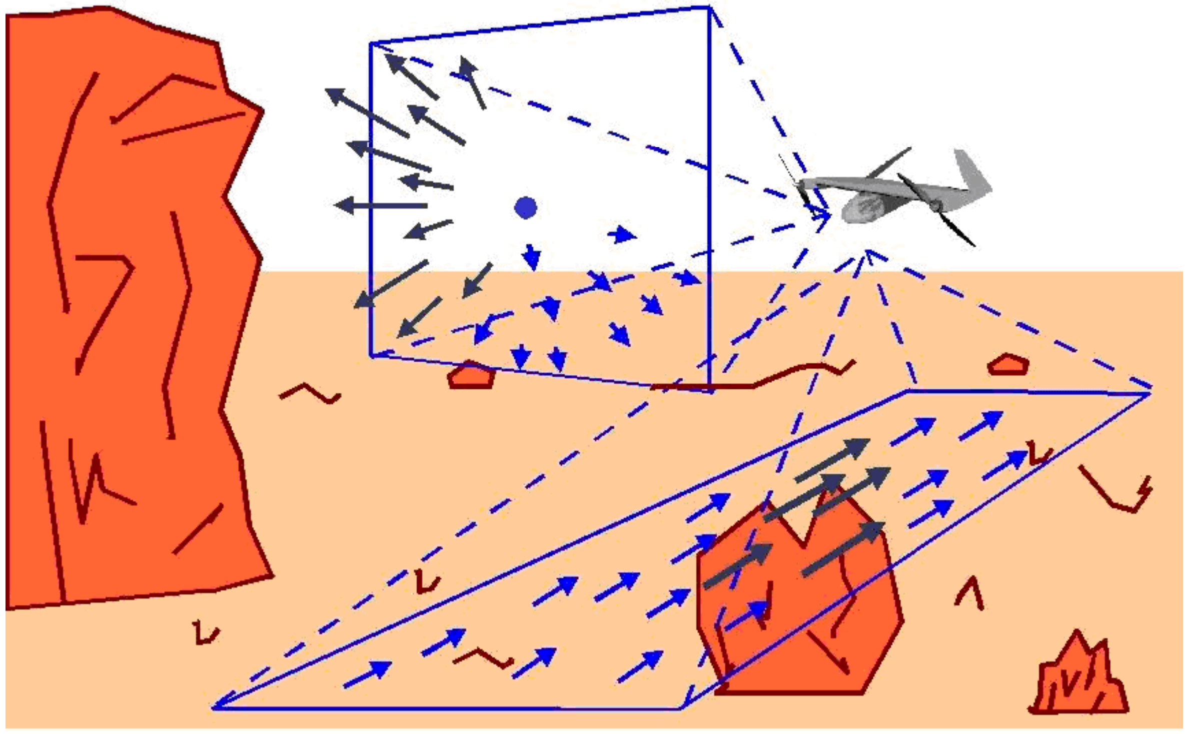
- **Phase 1**: Optical Flow Verfahren [Kourogi]
  - schnell: 5-50 fps
- **Phase 2**: Kritik an Optical Flow
  - große Probleme mit Real-World Endoskopmaterial
- **Phase 3**: Logarithmic Search
  - drastische Verbesserung der Stabilität
  - auch schnell (5-30 fps)

Optischer Fluss

Nehmen wir an, dass wir uns in einer statischen Welt befinden.

Bewegt sich ein Beobachter (eine Kamera) durch diese statische Welt, so entsteht auf seiner Retina (auf dem Kameratarget) dennoch Bewegung, der sog. "Optische Fluss" (engl. *optical flow*). Abbildung 1 links zeigt ein Beispiel für den Optischen Fluss aus der realen Welt.

Abbildung 1: Optischer Fluss (O.F.), den ein Vogel, Insekt oder ein Flugzeug sieht. Der blaue Punkt vorne ist der FOE (focus of expansion), rechts davon ist der O.F. größer aufgrund des nahen Felsens.



Optischer Fluss

- weitere Details s. TR-OpticalFlow-short.doc
  - Pseudo Motion, Akzeptanztest
  - Compensated Motion
  - LS-Schätzung der affinen Transformation

Algorithmus Optischer Fluss

- while (stopCondition != true)
  - Pseudo Motion auf Basis der Compensated Motion berechnen
  - Akzeptanztest für jedes Pixel
  - LS-Schätzung auf Grundlage akzeptierter Pixel → neues affines Motion Field (neue Compensated Motion)

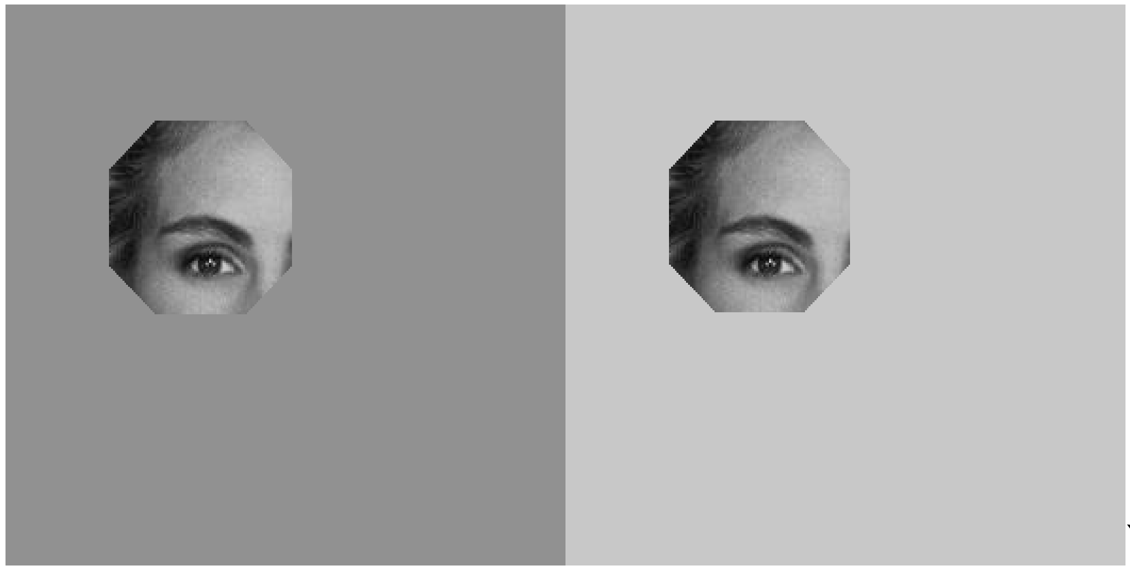
- Als stopConditions kommen in Frage
  - Anzahl Iterationen
  - Akzeptanztest für mehr als X% aller Pixel erfüllt
  - u.a.m.

Überblick

- **Phase 1**: Optical Flow Verfahren [Kourogi]
  - schnell: 5-50 fps
- **Phase 2**: Kritik an Optical Flow
  - große Probleme mit Real-World Endoskopmaterial
- **Phase 3**: Logarithmic Search
  - drastische Verbesserung der Stabilität
  - auch schnell (5-30 fps)

Kritik an Optical Flow

- funktioniert zwar prima für synthetische Sequenzen



- funktioniert leider kaum für reale Bildsequenzen
  - insbesondere nicht bei Endoskopbildern

Aktivierung: Wieso wohl?

FProjekt-BV-3D-Endo\endo_movie\movie-jroberts3.gif
<endo_movie>\Endo in Mpeg1

Kritik an Optical Flow

Was sind die Probleme?

- Beleuchtung, Beleuchtung, Beleuchtung!
  - globaler Kontrast
  - globale Helligkeit
  - lokale Beleuchtungsunterschiede: Beleuchtungsgradient zum Rand hin
  - jedes dieser Problem zerstört mit hoher Wahrscheinlichkeit den Akzeptanztest im Kourogi-Algo
- einzelne Pixel, die O.F. vergleicht, tragen zu wenig Struktur
- deshalb: ersetze Akzeptanztest, der auf Ein-Pixel-Grauwert beruht, durch strukturtragenderen Vergleich

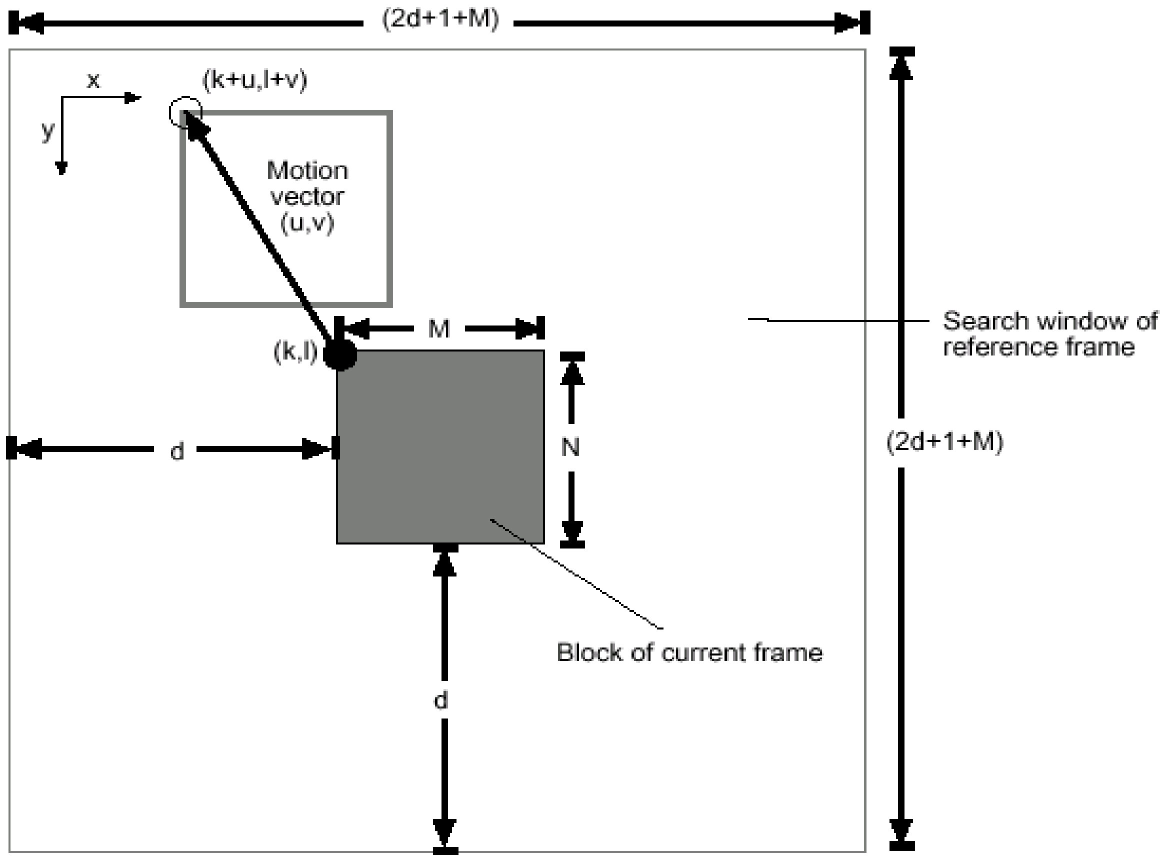
FProjekt-BV-3D-Endo\endo_movie\movie-jroberts3.gif
<endo_movie>\Endo in Mpeg1

Überblick

- **Phase 1**: Optical Flow Verfahren [Kourogi]
  - schnell: 5-50 fps
- **Phase 2**: Kritik an Optical Flow
  - große Probleme mit Real-World Endoskopmaterial
- **Phase 3**: Logarithmic Search
  - drastische Verbesserung der Stabilität
  - auch schnell (5-30 fps)

  - Logarithmic Search basiert auf **Block Matching**, das wir nachfolgend kurz vorstellen

EXHAUSTIVE SEARCH



- Simplest algorithm, but computationally most expensive
- Evaluates cost function (see next slide) at every location in the search area
- Cost function is evaluated K=(2d+1)^2 times.
- K(d=7) = 15*15 = **225 iterations**.

from A. Girotra, „Motion Estimation
http://sitemaker.umich.edu/agirotra/files/SANPre.ppt

Block Comparision: Cost Functions

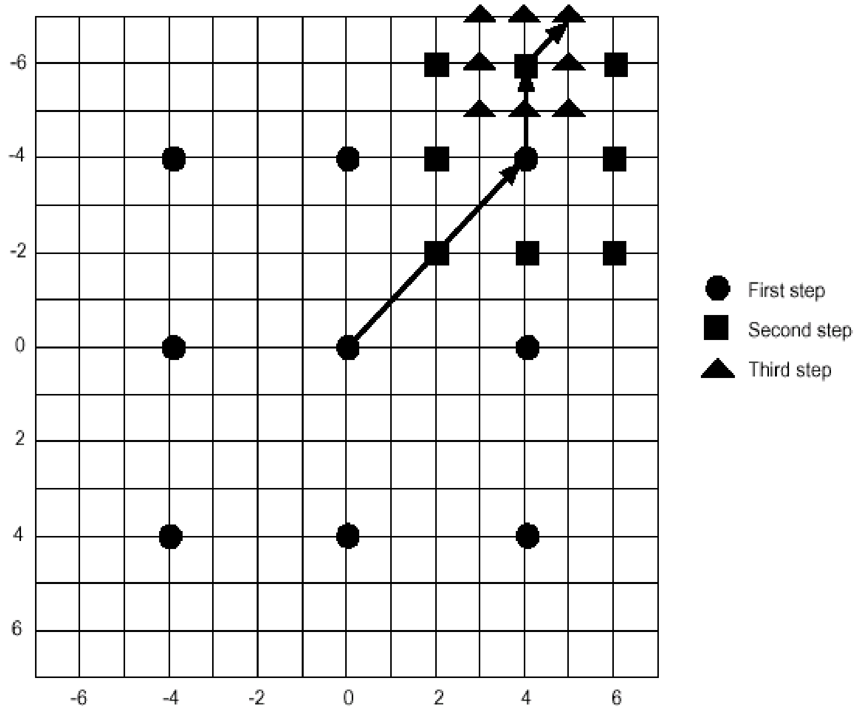
Given two blocks with N*M=n pixels each. How can we compare the similarity of the grey values in these blocks? Some examples:

- MSE: mean square error
- MAE: mean absolute error
- SAD: sum of absolute differences (=n*MAE)
- CCF: cross correlation function

$$\frac{1}{n-1}\sum_{x,y}\frac{(f(x,y)-\overline{f})(t(x,y)-\overline{t})}{\sigma_f\sigma_t}$$

Activation: Advantage of CCF?

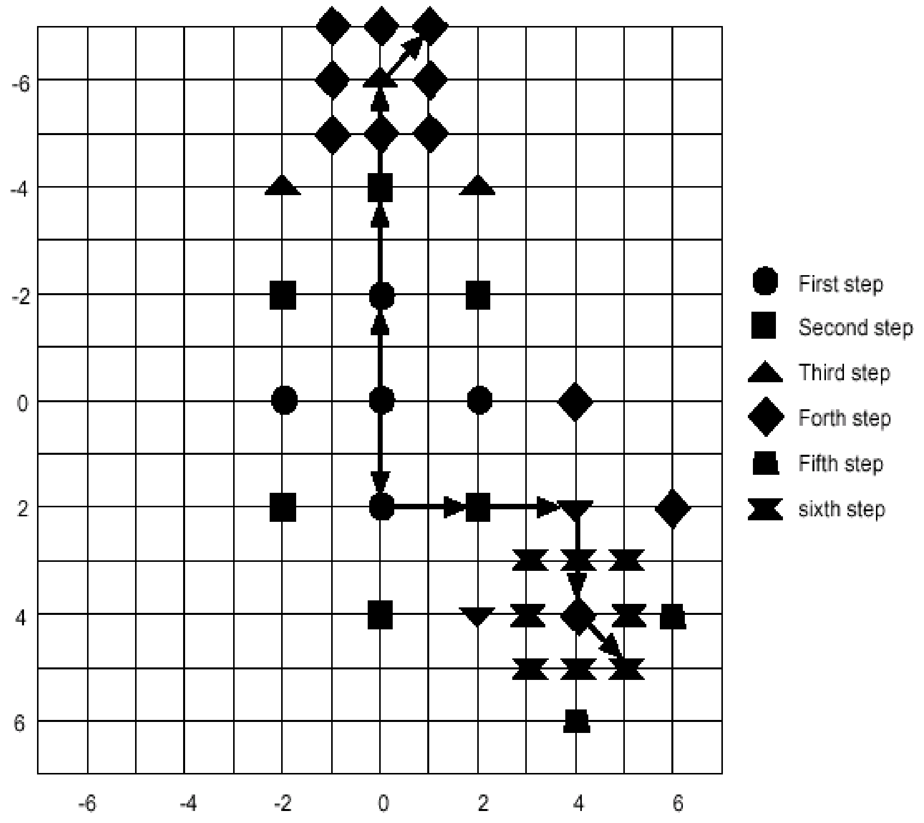
THREE STEP SEARCH (3SS)



The three-step search algorithm (3SS) is proposed by Koga et. al. in 1981 [1]. This algorithm is based on a coarse-to-fine approach with logarithmic decreasing in step size as shown. The initial step size is half of the maximum motion displacement d .

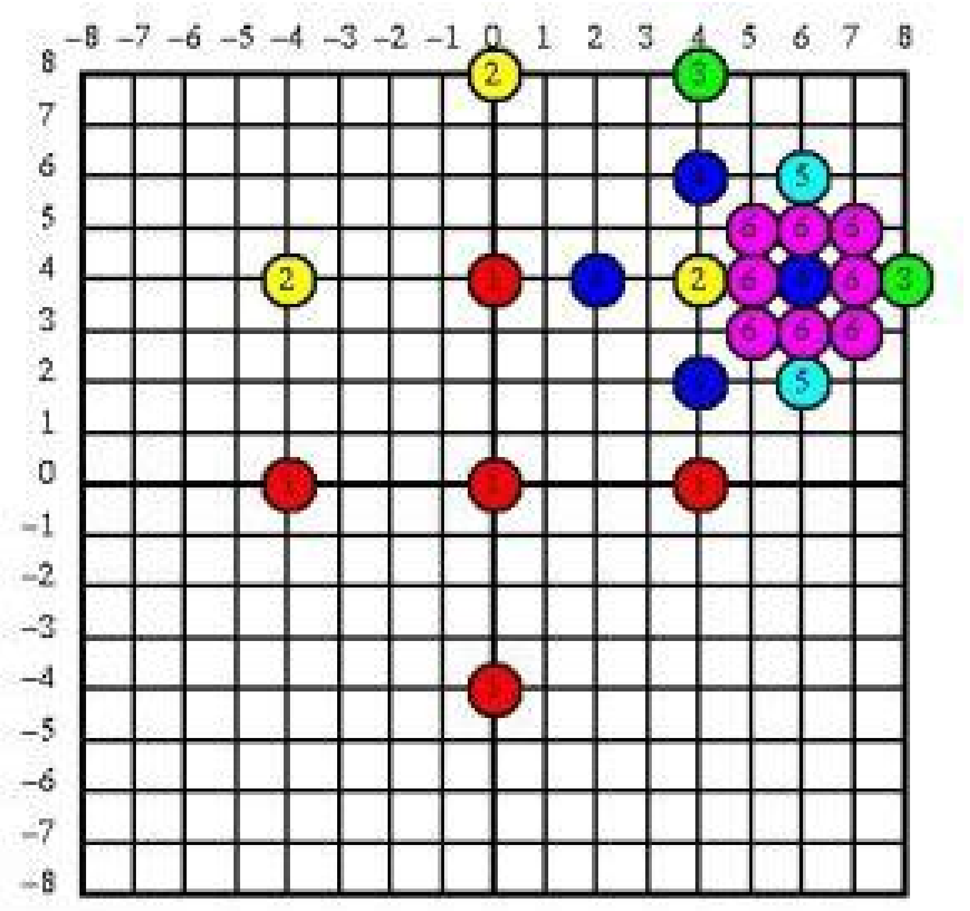
For each step, nine checking points are matched and the minimum point of that step is chosen as the starting center of the next step. For d = 7, the number of checking points required is (9 + 8 + 8)=**25 iterations**. For larger search window (i.e. larger d), 3SS can be easily extended to n-steps using the same searching strategy with the number of checking points required equals to [1 + 8 log2(d + 1)].

2D LOGARITHMIC SEARCH



2D-logarithmic search (2DLOG) is proposed by Jain et. al. in 1981 [8]. It uses a (+) cross search pattern in each step. The initial step size is [d/4]
The step size is reduced by half only when the minimum BDM point of previous step is
the center one or the current minimum point reaches the search window boundary.
Otherwise, the step size remains the same. When the step size reduced to 1, all the 8 checking points adjacent to the center checking point of that step are searched. Two different search paths are shown. The top search path requires (5 +3 +3 +8) = 19 checking points. The lower-right search path requires (5+3+2+3+2+8) =23 checking points.

2D LOGARITHMIC SEARCH
(another example)



2D-logarithmic search with d=8, initial step size is [d/2]
Required evaluations in this example:
(5+3+2+4+2+8)= **24 cross correlations**.

Variant used in ImageMosaicing:
- not 5-point cross, but 9-point neighborhood
- WHY?

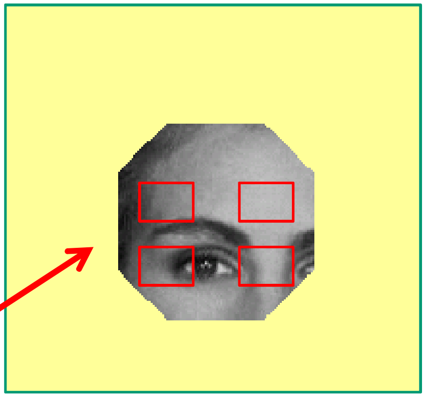
Activation:
When may
CCF fail?

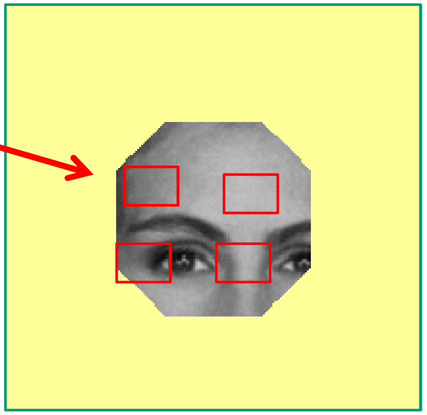
2D Logarithmic Search
Pros and Cons

- Pros
  - LogSearch has all advantages of CCF:
    - Invariance against brightness shift
    - Invariance against contrast change
    - Both globally AND locally
  - LogSearch much faster than Exhaustive Search
- Cons
  - CCF is expensive (computation time)
  - LogSearch can fail (miss the global minimum, which would be found by Exhaustive Search)
    - $\rightarrow$ therefore we use 9-point neighborhood
    - Side remark: LogSearch guaranteed to find the global minimum, if CCF is convex (has only one minimum in [±d, ±d])
  - Both LogSearch and Exhaustive Search will not find the true match if it is outside search range [±d, ±d] $\rightarrow$ we need initial estimate

Motion Field Estimation with LogSearch
The Algorithm

- Step 0: Translation estimate with O.F. (starting point for LogSearch)
- Step 1: Place N*N blocks on image region
- Step 2: Locate them in the new frame with CCF LogSearch (starting from Step 0-estimate). Accept the K best blocks and those with CCF≥ c_0.
- Step 3: LS estimate of affine global motion $\rightarrow$ dense vector field (u_c,v_c)





Applications for Motion Field Estimation

- Image Mosaicing (see PanoStream2.avi)
- Image Registration
- Noise Reduction (by superimposing a set of noisy images)
- Superresolution
  - small objects become „fuzzy“ due to pixel quantization
  - superimpose different frames and calculate grey values at a finer resolution level
  - see r004\sequences and r016\sequences
  - (it doesn‘t work always: r014\sequences)

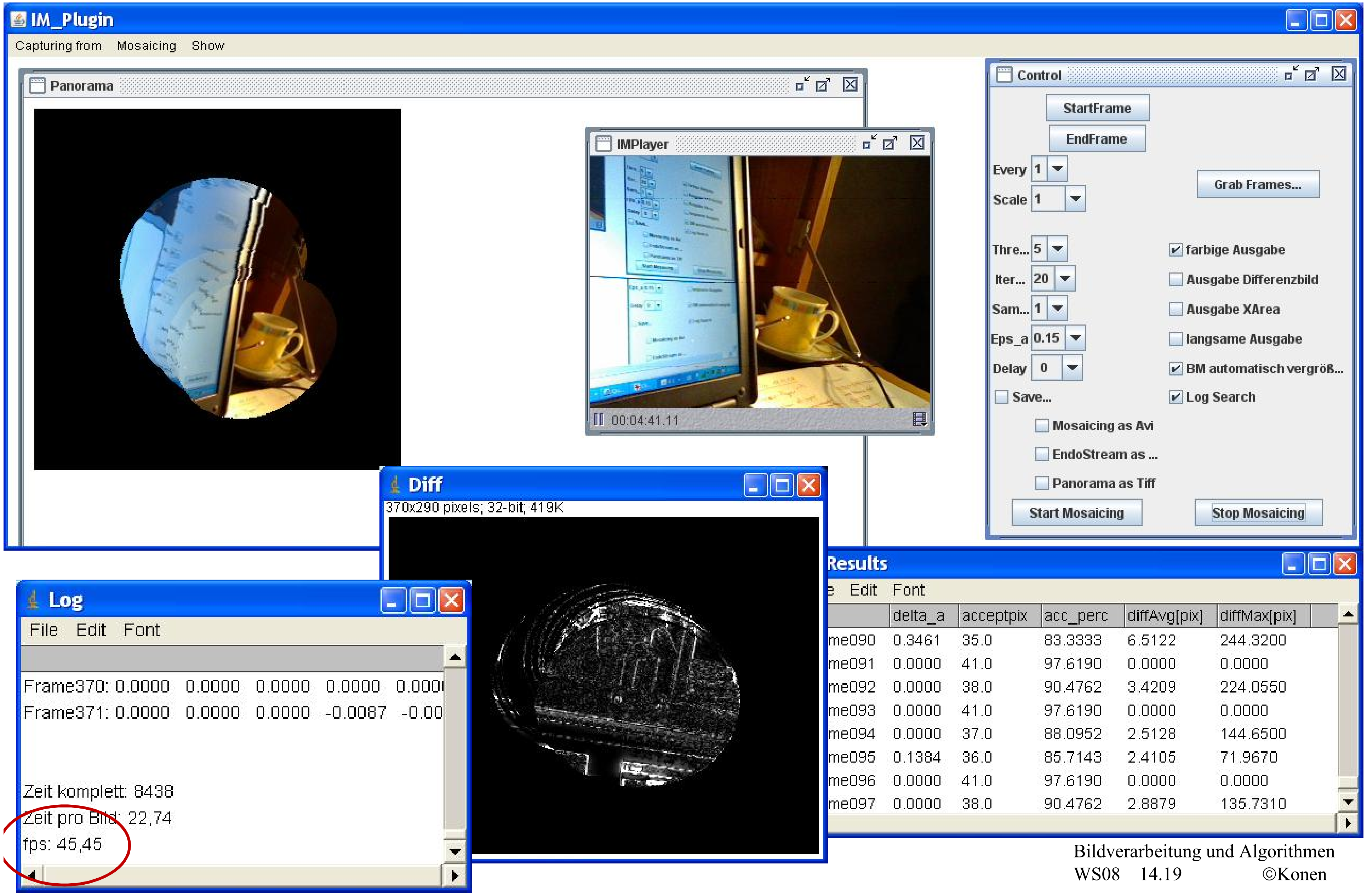
Demo

- Demonstration of application ***RealMosaic***
  - in clinical research for neuroendoscopy
  - fully automated image mosaic, no user intervention needed
  - very robust against local illumination changes

Demo

- Experiment with it, using the ‚live' camera stream
  - Observe the strong limitations of OpticalFlow algo!
  - Now compare this with LogSearch
  - Find out where the current limitations of LogSearch are. Try to describe them!

Demo – Some Screenshots



Every 1
Scale 1
Thre... 5
Iter... 20
Sam... 1
Eps_a 0.15
Delay 0

[✓] farbige Ausgabe
[ ] Ausgabe Differenzbild
[ ] Ausgabe XArea
[ ] langsame Ausgabe
[✓] BM automatisch vergröß...
[✓] Log Search
[ ] Save...
[ ] Mosaicing as Avi
[ ] EndoStream as ...
[ ] Panorama as Tiff

370x290 pixels; 32-bit; 419K

```
Frame370: 0.0000   0.0000   0.0000   0.0000   0.000
Frame371: 0.0000   0.0000   0.0000   -0.0087   -0.00


Zeit komplett: 8438
Zeit pro Bild: 22,74
fps: 45,45
```

	delta_a	acceptpix	acc_perc	diffAvg[pix]	diffMax[pix]
me090	0.3461	35.0	83.3333	6.5122	244.3200
me091	0.0000	41.0	97.6190	0.0000	0.0000
me092	0.0000	38.0	90.4762	3.4209	224.0550
me093	0.0000	41.0	97.6190	0.0000	0.0000
me094	0.0000	37.0	88.0952	2.5128	144.6500
me095	0.1384	36.0	85.7143	2.4105	71.9670
me096	0.0000	41.0	97.6190	0.0000	0.0000
me097	0.0000	38.0	90.4762	2.8879	135.7310

Weitere Themenangebote

- Themenangebote ***RealMosaic***

siehe Website

http://www.gm.fh-koeln.de/~konen/Diplom+Projekte/FProjekt-BV-3D-Endo

- Anschauungsmaterial ***Objektregistrierung / Superresolution***
  C:\Backup-WK\DVD.Jannis.Bloemendal\ergebnisse\

Literature

1. [Koga81] T. Koga et.al. “Motion Compensated interframe coding for video conferencing,” *Proc. Nat. telecom. Conf., New* Orleans,LA,pp.G5.3.1-G5.3.5 , 1981
2. [Jain81] J.R. Jain and A.K. Jain, “Displacement measurement and its application in interframe image coding,” *IEEE Trans. Commun., Vol. COM-29, pp. , 1799-1808,* Dec. 1981
3. [Kourogi99] : M. Kourogi, T. Kurata, J. Hoshino, and Y. Muraoka. *Real-time image mosaicing from a video sequence*. In Proc. ICIP99,vol. 4, 133--137, 1999. http://citeseer.ist.psu.edu/253440.html.
4. [HornSchunck81] B. K. P. Horn and B. G. Schunck, *Determining optical flow*, Artificial Intelligence, **17**, 1-3, pp. 185--203, 1981.