Image Registration
Lecture 13: Robust Estimation

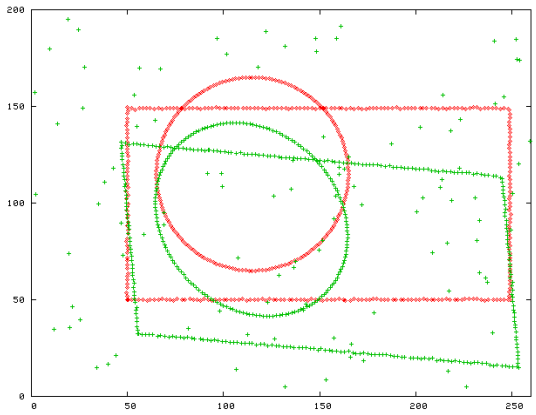
Prof. Charlene Tsai

Today's Lecture

- Motivating problem:
  - Mismatches and missing features
- Robust estimation:
  - Reweighted least-squares
  - Scale estimation
- Implementation in rgrl
- Solving our motivating problem

Motivation

- What happens to our registration algorithm when there are missing or extra features?
- Mismatches tend to occur, causing errors in the alignment
- Our focus today is how to handle this situation



3

Robust Estimation

- Down-grade or eliminate the “influence” of mismatches (more generally, gross errors or “outliers”)
- Base the estimate of the transformation on correct matches (“inliers”)
- Major question is how to distinguish inliers from outliers
  - Studying ways to answer this question has occupied statisticians and computer vision researchers for many years.
  - We’ll look at just a few of the answers.

4

A Simplified Problem: Location Estimation

- Given:
  - A set of *n* scalar values $\{x_i\}$
- Problem:
  - Find the mean and variance (or standard deviation) of these values
- Standard solutions:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

5

Maximum Likelihood Estimation

- The calculation of the average from the previous slide can be derived from standard least-squares estimation, which in turn derives from Maximum-Likelihood Estimation
- The (normal) probability of a particular random x_i value:

$$p(x_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

- The probability of *n* independent random values:

$$p(x_1, \ldots, x_n; \mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

- The negative log likelihood:

$$-\log p(x_1, \ldots, x_n; \mu, \sigma^2) = \frac{n}{2}\log 2\pi + \frac{n}{2}\log\sigma^2 + \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

Estimating the Mean

- Focusing only on μ, we can ignore the first two terms, and focus on the least-squares problem of minimizing:

$$\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2}$$

Remember this

- Computing the derivative with respect to μ and setting it equal to 0 yields:

$$\sum_{i=1}^{n} \frac{x_i - \mu}{\sigma^2} = 0$$

- Solving leads to the original estimate of the mean:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

7

Estimating the Variance

- Substituting the estimate into the log-likelihood equation and dropping the terms that don't depend on the variance produces:

$$\frac{n}{2} \log \sigma^2 + \sum_{i=1}^{n} \frac{(x_i - \hat{\mu})^2}{2\sigma^2}$$

- Computing the derivative with respect to σ

$$\frac{n}{\sigma} - \frac{1}{\sigma^3} \sum_{i=1}^{n} (x_i - \hat{\mu})^2$$

- We solve for the variance by setting this equal to 0 and rearranging:

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \hat{\mu})^2$$

8

What About Outliers?

- When one outlier is introduced in the data, it skews the least-squares (mean) estimate.
- If we move the position of this outlier off toward infinity, the estimate follows it.

9

What About Outliers?

- When one outlier is introduced in the data, it skews the least-squares (mean) estimate.
- If we move the position of this outlier off toward infinity, the estimate follows it.

10

What About Outliers?

- When one outlier is introduced in the data, it skews the least-squares (mean) estimate.
- If we move the position of this outlier off toward infinity, the estimate follows it.

11

What About Outliers?

- When one outlier is introduced in the data, it skews the least-squares (mean) estimate.
- If we move the position of this outlier off toward infinity, the estimate follows it.

12

Why Is This?

- The least-squares "cost function" is quadratic:

$$\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{\sigma^2}$$

- Or, stated another way, the estimate changes (slowly) with the change in any one point, no matter how far away that point is:

$$\frac{\partial \hat{\mu}}{\partial x_n} = \frac{\partial}{\partial x_n} \frac{1}{n} \sum_{i=1}^{n} x_i = \frac{1}{n}$$

13

M-Estimators: A Different Cost Function

- One solution is to replace the least-squares cost function with one that does not grow quadratically for large errors:

$$\sum_{i=1}^{n} \rho\left(\frac{x_i - \mu}{\sigma}\right)$$

- As examples:

$$\rho(u) = \begin{cases} (1 - u^2/c^2)^3 & |u| \le c \\ 1 & |u| > c \end{cases}$$

Beaton-Tukey biweight

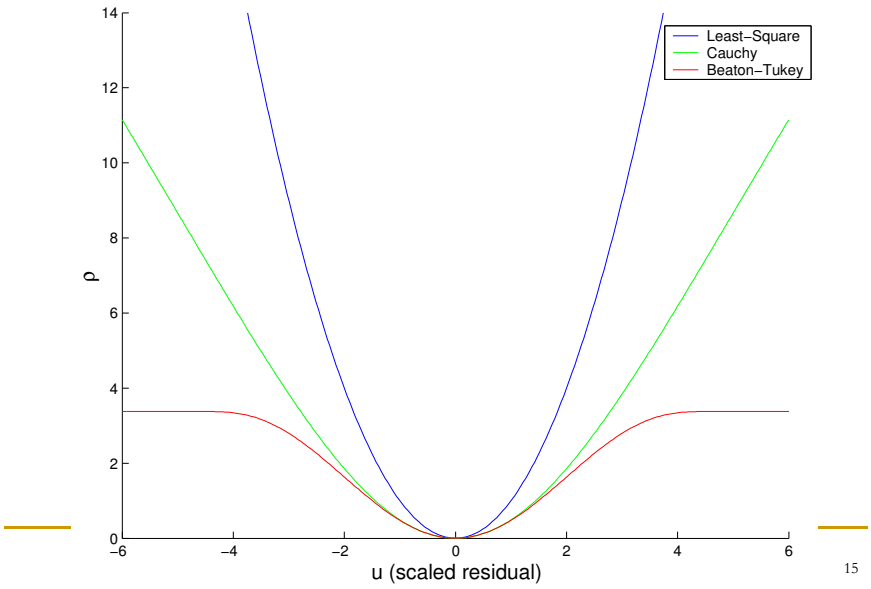
$$\rho(u) = \log(1 + u^2/c^2)$$

Cauchy

where $u = \frac{x_i - \mu}{\sigma}$

14

Plots of M-Estimator Shapes



15

Issues

- Non-linear estimation due to non-quadratic cost function
- Dependence on σ

$$u = \frac{x_i - \mu}{\sigma}$$

  - u is the “scaled normalized residual”
  - Intuitively, think of u=3 as “3 standard deviations”, which is on the tail of a normal distribution
  - Warning: requires a good estimate of σ
- Tuning constant c:
  - For Beaton-Tukey is usually in range 4.0 - 5.0
  - For Cauchy, c is usually in range 1.0 - 3.0
  - Tune by thinking about weights at u=2, 3, 4…

16

Minimization Techniques

- Our goal is to estimate μ by minimizing

$$\sum_{i=1}^{n} \rho\left(\frac{x_i - \mu}{\sigma}\right)$$

  For the moment we will assume σ is known.
- Gradient descent or Levenberg-Marquardt techniques can be used.
- Many implementations of M-Estimators, however, including ours, use a technique known as re-weighted least-squares (IRLS).

17

Derivation(1)

- Let the function to be minimized be

$$E(\mu) = \sum_{i=1}^{n} \rho\left(\frac{x_i - \mu}{\sigma}\right)$$

- Take the derivative with respect to μ

$$\frac{\partial E}{\partial \mu} = \sum_{i=1}^{n} \rho'\left(\frac{x_i - \mu}{\sigma}\right)\left(\frac{-1}{\sigma}\right) = \sum_{i=1}^{n} \psi\left(\frac{x_i - \mu}{\sigma}\right)\left(\frac{-1}{\sigma}\right)$$

- Define a new function $w(u) \cdot u = \psi(u)$
- Substitute into the above, set equal to 0, and multiply by $-\sigma^2$:

$$\sum_{i=1}^{n} (x_i - \mu) w\left(\frac{x_i - \mu}{\sigma}\right) = 0$$

18

Derivation (2)

- In this equation, we have two terms that depend on μ. Let's define

$$w_i = w\left(\frac{x_i - \mu}{\sigma}\right)$$

- and substitute it in:

$$\sum_{i=1}^{n}(x_i - \mu)w_i = 0$$

- Momentarily keeping w_i fixed, we can solve for μ:

$$\hat{\mu} = \frac{\sum w_i x_i}{\sum w_i}$$

The weighted least-squares estimate

19

Derivation (3)

- But, w_i depends on μ, which depends on w_i…
- So, we iterate:
  - Fix μ and calculate w_i
  - Fix w_i and calculate the new estimate for μ
- Sound familiar?
  - It has an iterative structure similar to that of ICP
- Of course, this requires an initial estimate for μ
- This is the "Iteratively Reweighted Least Squares" procedure

20

The Weight Functions

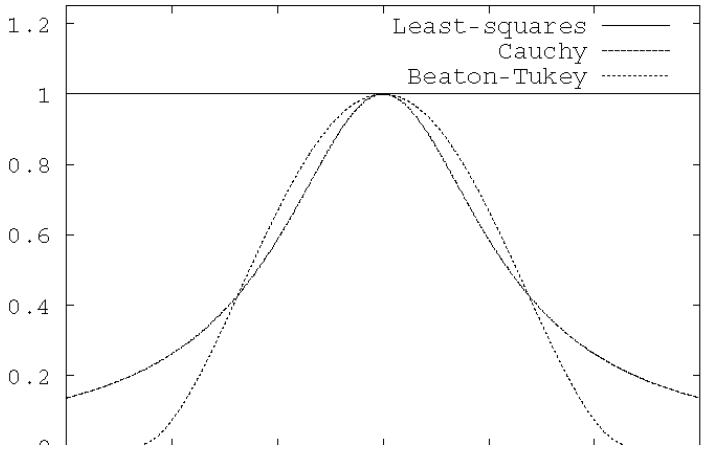
- The weight functions control the amount of influence a data value has on the estimate
- When the weight is $w_i = 1$, independent of μ, we are back to least-squares
- Here are the other weight functions:

$$w(u) = \begin{cases} (1 - u^2/c^2)^2 & |u| \le c \\ 0 & |u| > c \end{cases}$$ Beaton-Tukey

$$w(u) = \frac{1}{1 + u^2/c^2}$$ Cauchy

21

The Weight Functions



Notice that the weight goes to 0 in the Beaton-Tukey function for $|u|>c$

22

Estimates of Scale

- Options:
  - Use prior information (note that this will not help much when we get back to registration)
  - Estimate scale using weighted errors
  - Estimate scale using order statistics such as the median (unweighted scale)

23

Weighted Scale

- Calculate from the IRLS weights.
- There are two possibilities:

$$\hat{\sigma}^2 = \frac{\sum w_i (x_i - \hat{\mu})^2}{n}$$

- And $$\hat{\sigma}^2 = \frac{\sum w_i (x_i - \hat{\mu})^2}{\sum w_i}$$

- The first is more aggressive and tends to under-estimate scale.

24

Unweighted Scale

- For given estimate μ, calculate "absolute" errors:

$$e_i = |x_i - \hat{\mu}|$$

- Find the median of these errors

$$e_m = \text{median}\{e_i\}$$

- The scale estimate is $1.483 e_m$
- The multiplier normalizes the scale estimate for a Gaussian distribution. Small sample correction factors are also typically introduced.
- More sophisticated unweighted scale estimators exist in our robust estimation software library.

25

Putting It All Together

- Given initial estimate
- Compute initial scale estimate (unweighted) or use one that's given
- Do:
  - Compute weights
  - Estimate μ (weighted least-squares estimate)
  - In the first few iterations, re-estimate σ^2
    - Need to stop this to let the overall estimate converge
- Repeat until convergence
- *Note: Initialization is a significant issue.*
  - For location estimation, often initializing with just the median of the x_i is good enough

26

Back to Registration

- Least-squares error function for a given set of correspondences:

$$E(\mathbf{\Theta}; \mathcal{C}) = \sum_{(\mathbf{g}_k, \mathbf{f}_k) \in \mathcal{C}} \left[D(\mathbf{T}(\mathbf{g}_k; \mathbf{\Theta}), \mathbf{f}_k) \right]^2$$

- Becomes the robust error function:

$$E(\mathbf{\Theta}, \mathcal{C}) = \sum_{(\mathbf{g}_k, \mathbf{f}_k)} \rho(D(\mathbf{T}(\mathbf{g}_k; \mathbf{\Theta}), \mathbf{f}_k) \ / \ \sigma)$$

27

Mapping to the Registration Problem

- Error values from estimating the mean:

$$e_i = x_i - \mu$$

  become errors in the alignment of points:

$$r_k = D(\mathbf{T}(\mathbf{g}_k; \mathbf{\Theta}), \mathbf{f}_k)$$

- Here we have replaced the letter *e* with the more commonly used letter *r*, and will call these errors "residuals"
- We can calculate weights:

$$w_k = w\left(\frac{D(\mathbf{T}(\mathbf{g}_k; \mathbf{\Theta}), \mathbf{f}_k)}{\sigma} \right) = w(r_k / \sigma)$$

- We can estimate scale, e.g.: $\hat{\sigma}^2 = \frac{\sum w_k r_k^2}{\sum w_k}$

28

Mapping to the Registration Problem

- The weighted least-squares estimate

$$\hat{\mu} = \frac{\sum w_i x_i}{\sum w_i}$$

- Becomes (for a fixed set of correspondences) the weighted least-squares estimate

$$\underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{(\mathbf{g}_k, \mathbf{f}_k)} w_k D(\mathbf{T}(\mathbf{g}_k; \boldsymbol{\Theta}), \mathbf{f}_k)^2$$

- In short, for a fixed set of correspondences, we can apply IRLS directly, as long as we provide a way to calculate residuals and compute a weighted least-squares estimate

29

Mapping to ICP - Questions

- When we combine IRLS with ICP registration, some issues arise:
  - How should the two different loops be combined?
    - reweighting and re-estimation
    - rematching and re-estimation
  - When should scale be estimated?

30

Mapping To ICP - Procedural Outline

- Given initial transformation estimate
- Do
  - Compute matches
  - Estimate scale
  - Run IRLS with fixed scale and fixed set of matches:
    - Compute weights
    - Re-estimate transformation using weighted least-squares
  - Re-estimate scale
- Until converged

31

Software Toolkit - Robust Estimation

- `vxl_src/contrib/rpl/rrel/`
- `rrel_util.[h,txx]`
  - Implementation of weighted and unweighted scale estimators
  - Joint estimation of location and scale
- `rrel_irls.[h,cxx]`
  - Implementation of generic IRLS function
- `rrel_muse`
  - More sophisticated unweighted scale estimator
- See examples in
  - `vxl_src/contrib/rpl/rrel/examples`

32

Software Toolkit - Objects for Registration

- `rgrl_scale_estimator`: Estimate scale from matches using weighted and/or unweighted techniques
  - Base class, with subclasses for specific techniques
  - Includes possibilities for weighting based on criteria other than (geometric) error such as feature similarity
  - Not fully implemented
- `rgrl_scale`: Store the scale
  - Includes possibility of other a scale on similarity error

33

Software Toolkit - Objects for Registration

- `rgrl_weighter`
  - Computes the robust weight for each match based on the residual and the scale
  - Includes some advanced features
- Weights are stored back in the `rgrl_match_set`
- The `rgrl_estimator` accesses the weights from the `rgrl_match_set` during estimation

34

Example

- `registration_simple_shapes_outliers.cxx`
- Includes code for generating outliers
- We'll concentrate on the construction and use of the new objects in registration

35

main()

```
// Set up the feature sets
//
const unsigned int  dimension = 2;
rgrl_feature_set_sptr moving_feature_set;
rgrl_feature_set_sptr fixed_feature_set;
moving_feature_set = new rgrl_feature_set_location<dimension>(moving_feature_points);
fixed_feature_set =  new rgrl_feature_set_location<dimension>(fixed_feature_points);


// Set up the ICP matcher
//
unsigned int k = 1;
rgrl_matcher_sptr cp_matcher = new rgrl_matcher_k_nearest( k );

// Set up the convergence tester
//
double tolerance = 1.5;
rgrl_convergence_tester_sptr conv_test =
  new rgrl_convergence_on_weighted_error( tolerance );
```

36

main()

```
// Set up the estimator for affine transformation
//
int dof = 6;                    //parameter degree of freedom
int numSampleForFit = 3;        //minimum number of samples for a fit
rgrl_estimator_sptr estimator = new rgrl_est_affine(dof, numSampleForFit);


// Set up components for initialization
//
vector_2d x0(0,0);          //upper left corner
vector_2d x1(300,300);      //bottom right corner
rgrl_roi image_roi(x0, x1);

rgrl_transformation_sptr init_transform;
vnl_matrix<double> A(2,2);
A(0,0) = 0.996;  A(0,1) = -0.087;
A(1,0) = -0.087;  A(1,1) =0.996;
vector_2d t( 10, -13);
init_transform = new rgrl_trans_affine(A, t);
```

37

main()

```
// Set up the weighter
//
vcl_auto_ptr<rrel_m_est_obj>  m_est_obj( new rrel_tukey_obj(4) );
rgrl_weighter_sptr wgter = new rgrl_weighter_m_est(m_est_obj, false, false);


// Set up the scale estimators, both weighted and unweighted
//
int max_set_size = 1000;  //maximum expected number of features
vcl_auto_ptr<rrel_objective> muset_obj( new rrel_muset_obj( max_set_size , false) );

rgrl_scale_estimator_unwgted_sptr unwgted_scale_est;
rgrl_scale_estimator_wgted_sptr wgted_scale_est;

unwgted_scale_est = new rgrl_scale_est_closest( muset_obj );
wgted_scale_est = new rgrl_scale_est_all_weights();
```

38

main()

```
// Store the data in the data manager
//
rgrl_data_manager_sptr data = new rgrl_data_manager();
data->add_data( moving_feature_set,  // data from moving image
                fixed_feature_set,   // data from fixed image
                cp_matcher,          // matcher for this data
                wgter,               // weighter
                unwgted_scale_est,   // unweighted scale estimator
                wgted_scale_est);    // weighted scale estimator

rgrl_feature_based_registration reg( data, conv_test );
reg.set_expected_min_geometric_scale( 0.1 );
reg.set_max_icp_iter( 10 );
```

39

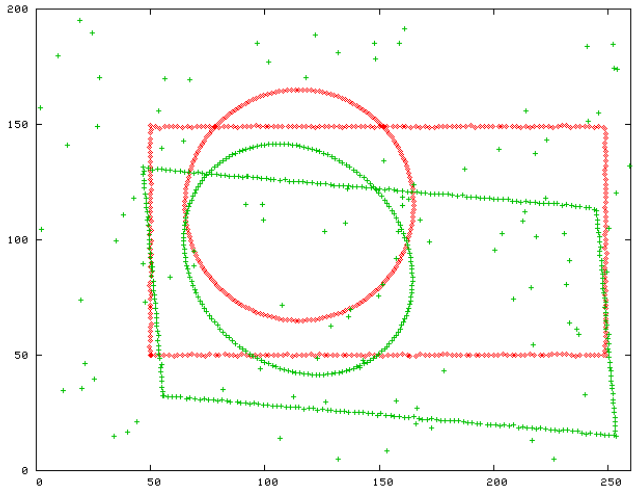
main()

```
// Run ...
//
reg.run( image_roi, estimator, init_transform );

if ( reg.has_final_transformation() ) {
  vcl_cout<<"Final xform: "<<vcl_endl;
  rgrl_transformation_sptr trans = reg.final_transformation();
  rgrl_trans_affine* a_xform = rgrl_cast<rgrl_trans_affine*>(trans);
  vcl_cout<<"Initial xform: A = "<<a_xform->A()<<vcl_endl;
  vcl_cout<<"Initial xform: t = "<<a_xform->t()<<vcl_endl;
  vcl_cout<<"Final alignment error = "<<reg.final_status()->error()<<vcl_endl;
}
```

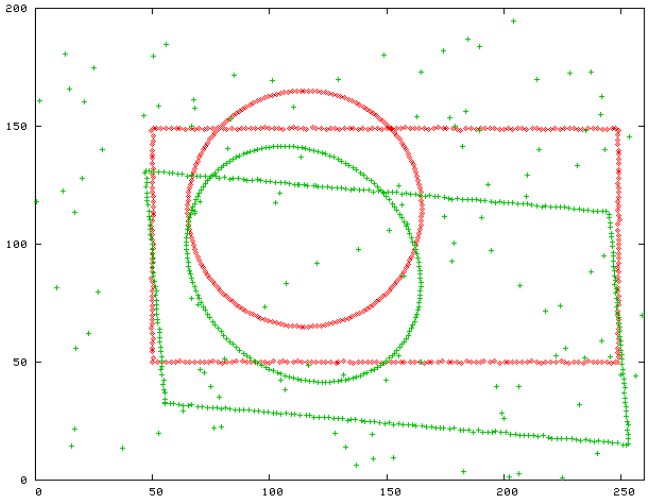
40

Example Revisited



41

Example Revisited



42

A Few More Comments on Scale Estimation

- Weighted and non-weighted scale estimators are used
  - Non-weighted is used first, before there is an initial scale estimate
  - Weighted is used, if it exists, after matching and scale have been initialized
- An advanced scale estimator, MUSE, is used for the initial unweighted scale
  - Can handle more than half outliers.

43

Summary

- Outliers arise from matches involving extraneous or missing structures
- Outliers corrupt the transformation estimate
- M-estimators can be used to down-weight the influence of outliers
- Scale estimation is a crucial part of robust estimation
- IRLS minimization is applied during registration for a fixed set of matches and a fixed scale estimate
- Robust estimation is tightly integrated in the registration toolkit

44