# Lecture 21
# Optimization Algorithms and Image Registration Wrap-Up

MP574: Applications

Sean B. Fain, PhD (sfain@wisc.edu)

Diego Hernando, PhD (dhernando@wisc.edu)

ITK/VTK Applications: Andrew Hahn, PhD (adhahn@wisc.edu)

# Learning Objectives

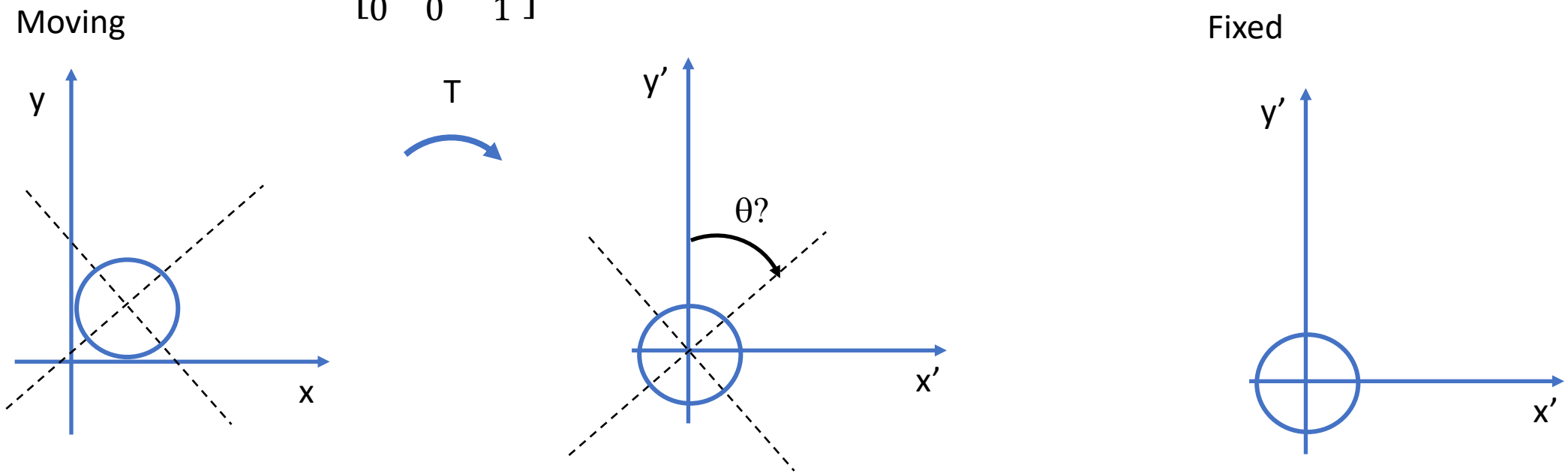
- Introduce the different optimization algorithms used for iterative image registration and their tradeoffs
- Review and synthesize the components for optimized image registration.
- Discuss examples from Homework 4 and their use of components of the iterative optimization approach: Transform, Interpolator, Cost Function, Optimization Algorithm.

# Image registration is ill-posed

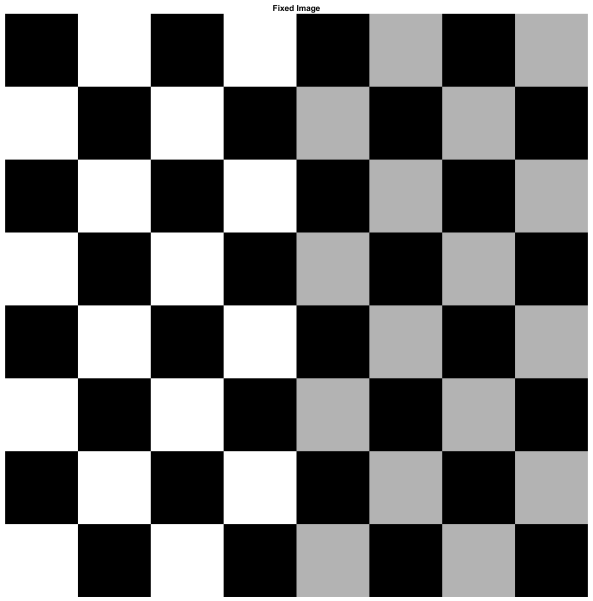
- Degrees of Freedom
  - The parameters estimated through the optimization strategy
  - Depends on the transformation used to model the motion

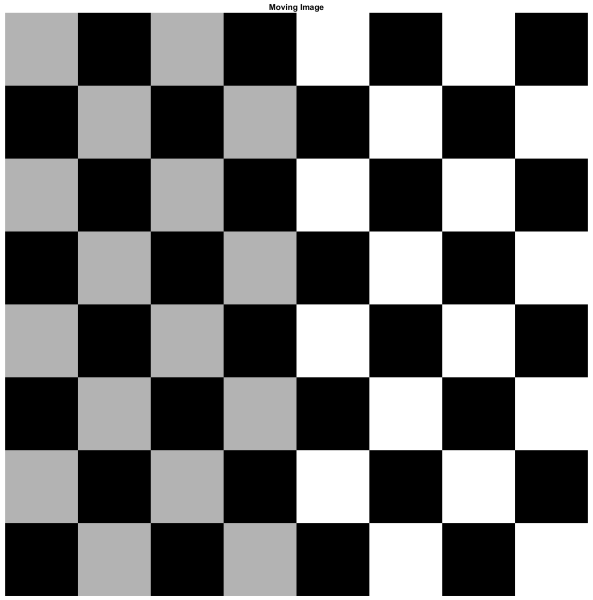
# Example – A simple 3 parameter estimation problem, where there is no unique solution

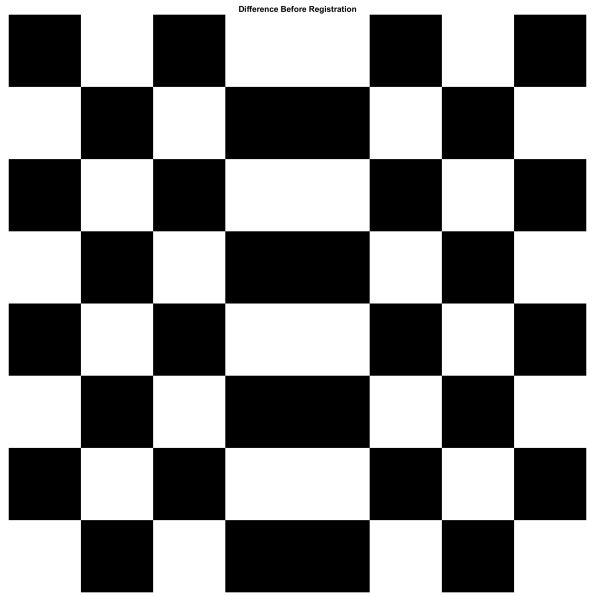
$$T = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$



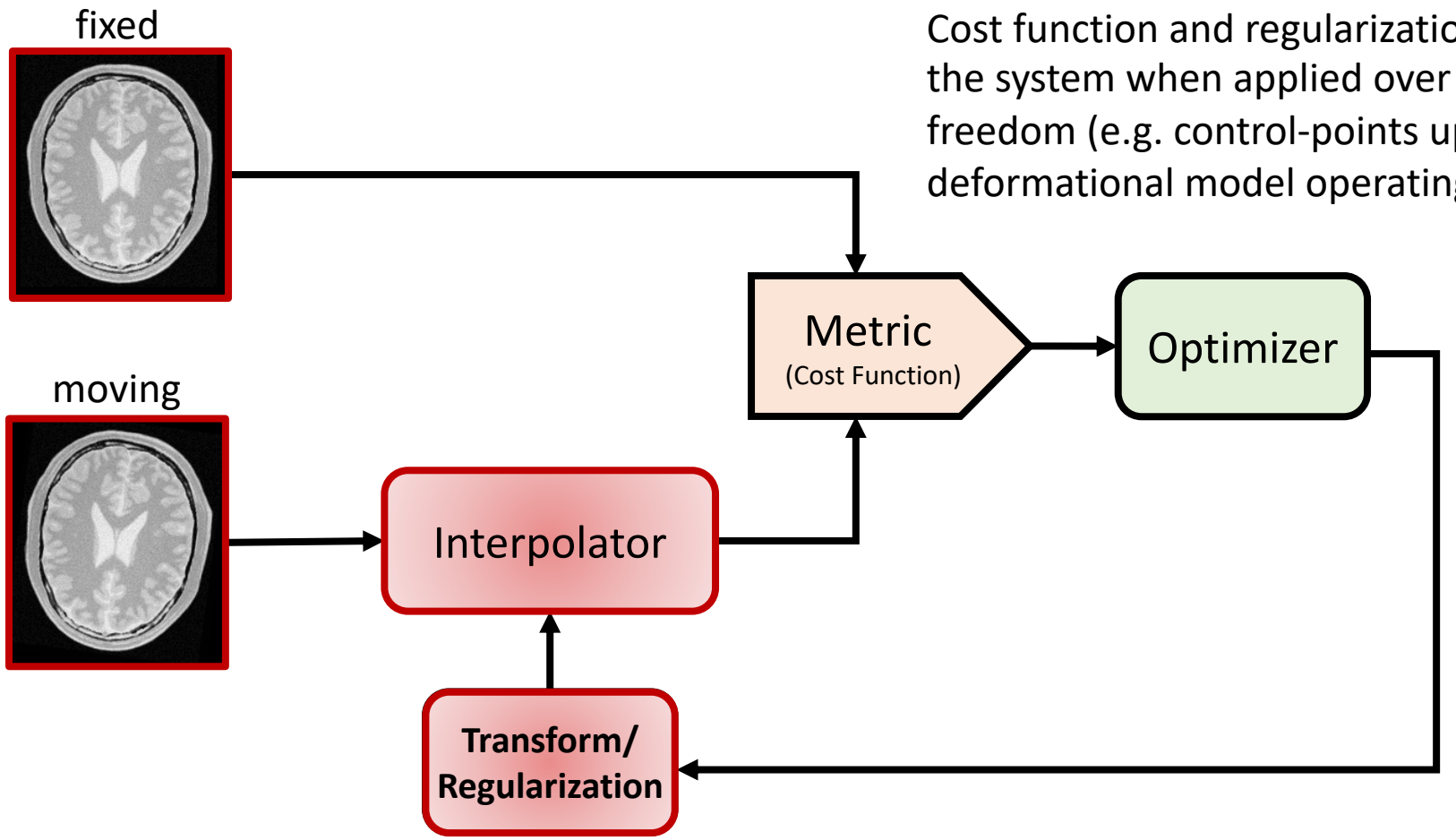
# Problem 5C: Highly symmetric structures



# Overview of Registration Workflow




Cost function and regularization reduce the dimensionality of the system when applied over the number of degrees of freedom (e.g. control-points up to a fully non-parametric deformational model operating on all voxels in the image).

# Mathematical Formulation of Optimized Iterative Image Registration

Recall the mathematical shorthand for the registration optimization problem from Lecture 19, Eq. 1.

$$M_{i+1} = C(F, T \circ M_i) + R(T)$$

- M ≡ “Moving” image
- F ≡ “Fixed” image
- $i$ ≡ current iteration index
- T ≡ transform operator
- C ≡ cost function
- R(T) ≡ regularization of the transform operation (usually a smoothing operation to minimize noise and interpolation error)

# Regularization

- Closely coupled with the transform.
- In many cases regularization is a smoothing operation that removes discontinuities and noise in the transform deformation field.
- Regularization may also serve to introduce prior knowledge regarding the solution such as placing constraints on the scale and direction (topological constraints) of the transform.
- In cases where the displacement of every image element (i.e., nonparametric deformation model)
  - Leads to high degrees of freedom
  - Only certain values of the Jacobian ($J>0$ or $J = 1$) allowed, for example.

# Optimization algorithm

At each iteration:

1. Set of transformation vectors are calculated,

$$\vec{r}_{i+1}(p) = \vec{r}_i(p) + \alpha_i \vec{g}_i\left(C\left(\vec{r}_i(p)\right) + R(\vec{r}_i(p))\right),$$

The last term is indexed in a simplified syntax as: $\vec{g}_i\left(\vec{r}_i(p)\right)$.

In words, $\vec{g}_i(\cdot)$ is the optimization algorithm that defines the search direction for each step of the iteration and operates on a feature space with dimensionality equal to the cost function as constrained by the regularized transform.

2. The term $\vec{g}_i\left(\vec{r}_i(p)\right)$ is calculated for each transform vector on the cost function and used to direct the next step of the transform to create an updated vector $\vec{r}_{i+1}(p)$.

# Choice of optimization algorithm depends on the registration problem and degrees of freedom

- Gradient descent and related methods
  - Perform well in the general case and are preferred in multi-modality registration problems.
  - Slower rates of convergence is the main tradeoff for gradient descent methods.
- Stochastic gradient descent is sometimes used to speed convergence
  - Typically works well in registration problems where the degrees of freedom are limited (i.e. B-spline deformable registration or lower).

# Gradient Descent Methods

| Algorithm | Formulation | Application |
|---|---|---|
| **Gradient descent** | $\vec{g}_i\left(\vec{r}_i(p)\right) = -\nabla_{\vec{r}_i(p)}(\vec{r}_i(p))$ | Multi-modality, non-parametric |
| **Conjugate gradient** | $\vec{g}_i(\vec{r}_i) = -\nabla_{\vec{r}_i}(\vec{r}_i) + \beta_i \vec{g}_{i-1}$ | Multi-modality, non-parametric, $\beta_i$ provides a scalar weighting of the previous gradient to accelerate convergence. |
| **Stochastic gradient descent** | $\vec{g}_i(\vec{r}_i) = \vec{r}_i(p) + \alpha_i \hat{\vec{g}}_i(\vec{r}_i(p))$ | Lower degree of freedom deformable registration transforms such as piecewise affine or B-spline. Several versions exist, here $\alpha_i$is used to indicate variable step-size with iteration, typically reducing with greater iterations to refine the search; $\hat{\vec{g}}_i$ approximates the gradient along randomized directions and for a subsample of uniformly sampled voxels. |

# Choice of optimization algorithm depends on the registration problem and degrees of freedom

- Gauss-Newton and Levenberg-Marquardt, are more robust in mono-modality registration problems.
- “Quasi-Newton” methods use the Jacobian to adjust the rate of convergence such that the transpose-square of the Jacobian approximates the second derivative of the local deformation.

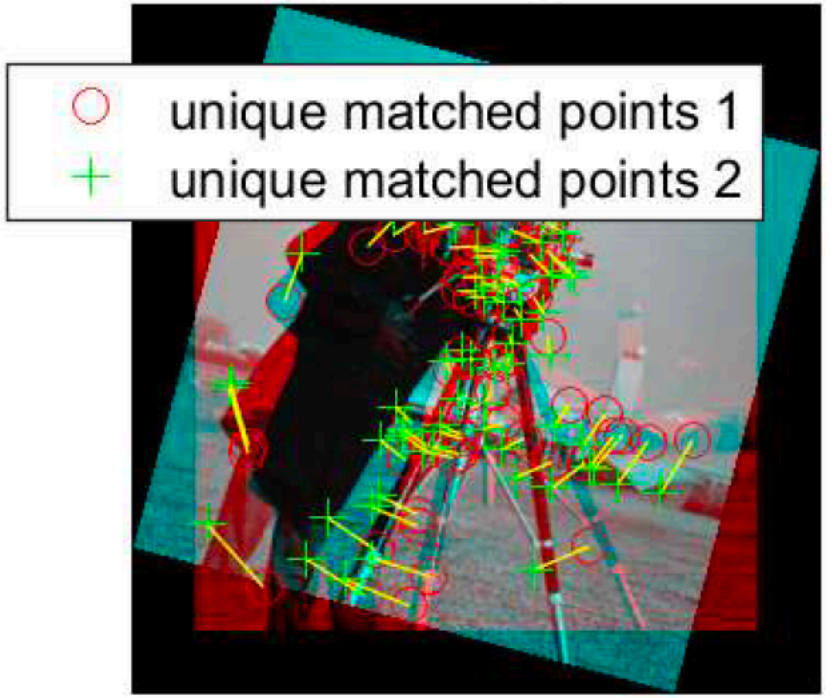
# Gauss-Newton Methods

| Algorithm | Formulation | Application |
|---|---|---|
| **Quasi-Newton** | $\vec{g}_i(\vec{r}_i) = -(J^T(\vec{r}_i)J(\vec{r}_i))^{-1}\nabla_{\vec{r}_i}(\vec{r}_i)$ | Mono-modality where $(J^T(\vec{r}_i)J(\vec{r}_i))$ approximates the change in local magnification and similarity of signal intensity is assumed across fixed and moving images. Thus, the gradient is weighted by the inverse of the local change in magnification (i.e. intuitively towards more trusted values). |
| **Levenberg-Marquardt** | $\vec{g}_i(\vec{r}_i) = -\left(H^{-1}(\vec{r}_i) + \xi\vec{I}\right)\nabla_{\vec{r}_i}(\vec{r}_i)$ | The $H^{-1}(\vec{r}_i)$ is the inverse of the Hessian matix, which in the case of Gaussian distributed signal intensities with similar variances, is equivalent to the covariance matrix. As we know from last semester, the diagonal elements of this matrix are the principal components of variance and represent the maximum curvature of the objective function in the corresponding directions. Note that $\vec{I}$ is the identity matrix, so that when the scalar $\xi$ is zero, the algorithm becomes the Gauss-Newton method. $\xi$ is used to adjust the rate of convergence. |

# Examples: Feature-Based Registration

**Difference Before Registration**

**Corresponding Features**



Fixed Image

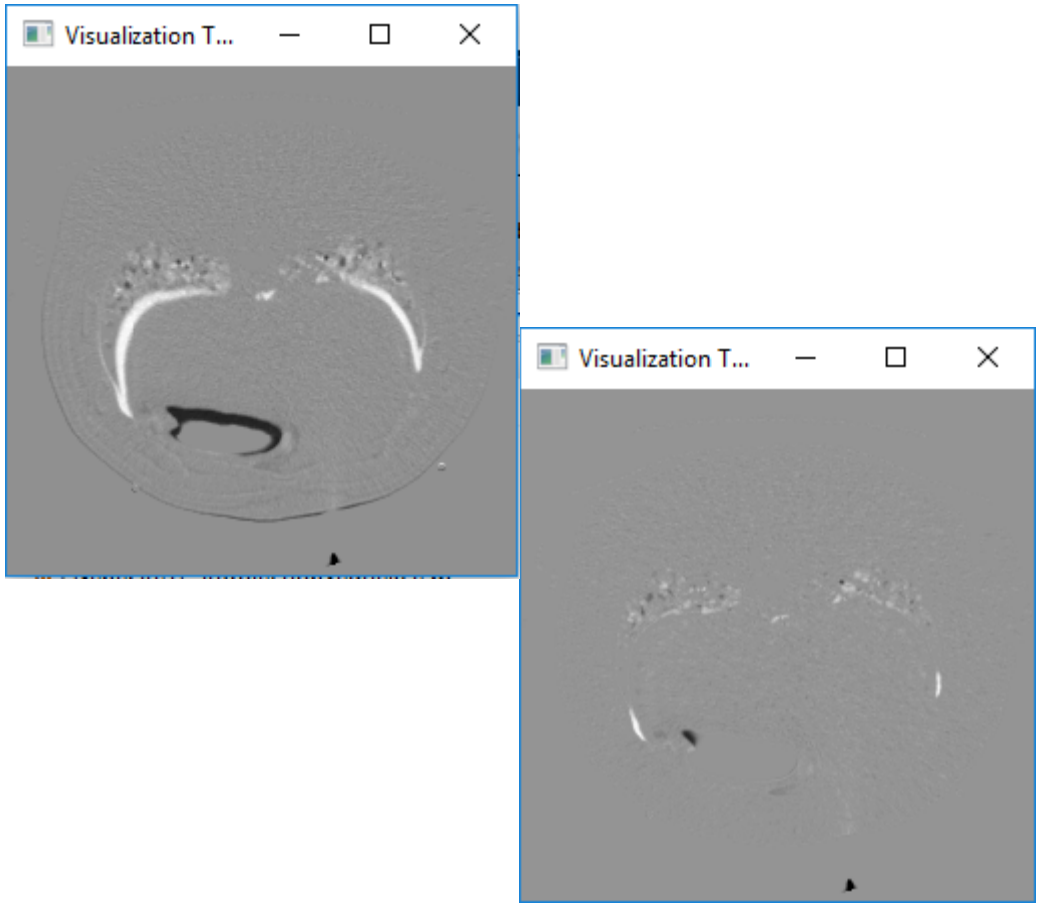
Registered Image

L1-Feature-Based Registered Result

_1-Feature-Based Registration Difference

# Exercise 4

1. Read moving and fixed image:
   - liver_exp.png
   - liver_insp.png
2. Perform histogram matching
3. Perform deformable registration and display status in each iteration
4. Display pre- and post registration results using difference filter
5. Display intermittent results during registration using the difference filter and VTK



# Deformable Registration

- itkDemonsRegistrationFilter
  - AddObserver(itk::IterationEvent(), ObserverType): Allows to perform track registration progress
  - SetFixedImage(ImageType): Fixed image
  - SetMovingImage(ImageType): Moving image
  - SetSmoothUpdateField(bool): Activate displacement field smoothing [true]
  - SetNumberOfIterations(int): Number of iterations [500]
  - SetStandardDeviations(double): Controls amount of smoothing [1.0]

# Preprocessing: Histogram Matching

- itk::HistogramMatchingImageFilter
  - SetInput(ImageType): Original moving image
  - SetReferenceImage(ImageType): Original fixed image
  - SetNumberOfHistogramLevels(int): Number of histogram bins [1024]
  - SetNumberOfMatchPoints(int): Number of match points [7]
  - ThresholdAtMeanIntensityOn()

```
class CommandIterationUpdate : public itk::Command
{
public:
    using Self = CommandIterationUpdate;
    using Superclass = itk::Command;
    using Pointer = itk::SmartPointer<Self>;
    itkNewMacro(Self);

protected:
    CommandIterationUpdate() {};

public:
    using OptimizerType = itk::OnePlusOneEvolutionaryOptimizer;
    using OptimizerPointer = const OptimizerType   *;

    void Execute(itk::Object *caller, const itk::EventObject & event) override
    {
      Execute((const itk::Object *)caller, event);
    }

    void Execute(const itk::Object * object, const itk::EventObject & event) override
    {
      auto optimizer = dynamic_cast< OptimizerPointer >(object);
      if (!itk::IterationEvent().CheckEvent(&event)) return;
      std::cout << optimizer->GetCurrentIteration() << optimizer->GetValue() <<
         optimizer->GetCurrentPosition() << std::endl;
    }
};
```

# One-Plus-One Evolutionary Optimizer

- An evolutionary algorithm iterates to find a set of parameters that produce the best possible registration result.
- Parent and child
  1. Perturbing, or mutating, the parameters from the last iteration (the parent).
  2. If the new (child) parameters yield a better result, then
  3. The child becomes the new parent whose parameters are perturbed, perhaps more aggressively.
  4. If the parent yields a better result, it remains the parent and the next perturbation is less aggressive.
- Biological model, but shares some features of conjugate gradient

# Summary

- General tradeoffs between gradient descent (multi-modality) and quasi-Gauss-Newton methods (mono-modality) are computational cost, and rate of convergence depending on the registration problem.
- Certain generalized approaches are commonly used because they are robust to multi-modality, converge in most situations, and mitigate interpolation errors.
  - NMI cost function
  - Conjugate gradient and bilinear or cubic spline interpolation
- The registration problem can be treated as a modular problem in which transform, cost function, and optimization algorithm can be chosen to be tailored to the problem of interest and with tradeoffs defined by the investigator.