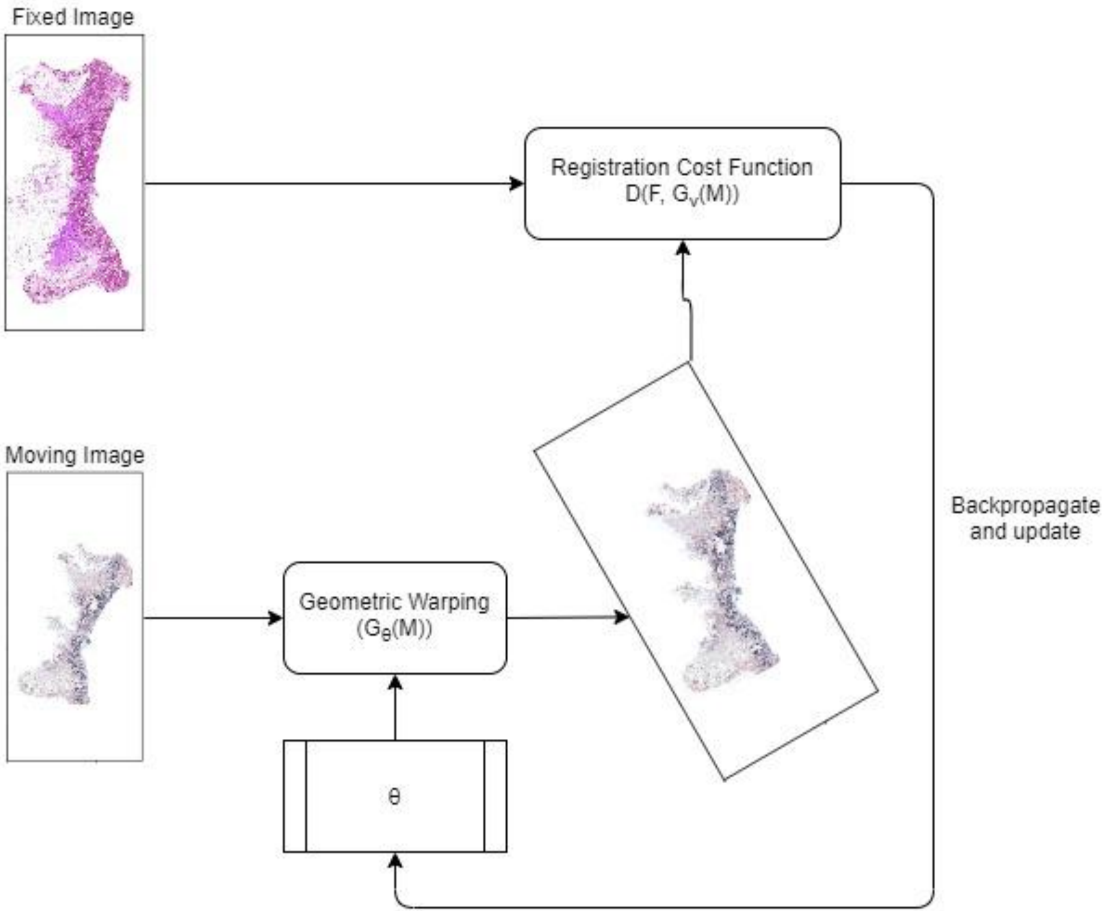
Differentiable Mutual Information and Matrix Exponential for Multi-Resolution Image Registration

Abhishek Nan, Matthew Tennant, Uriel Rubin, Nilanjan Ray

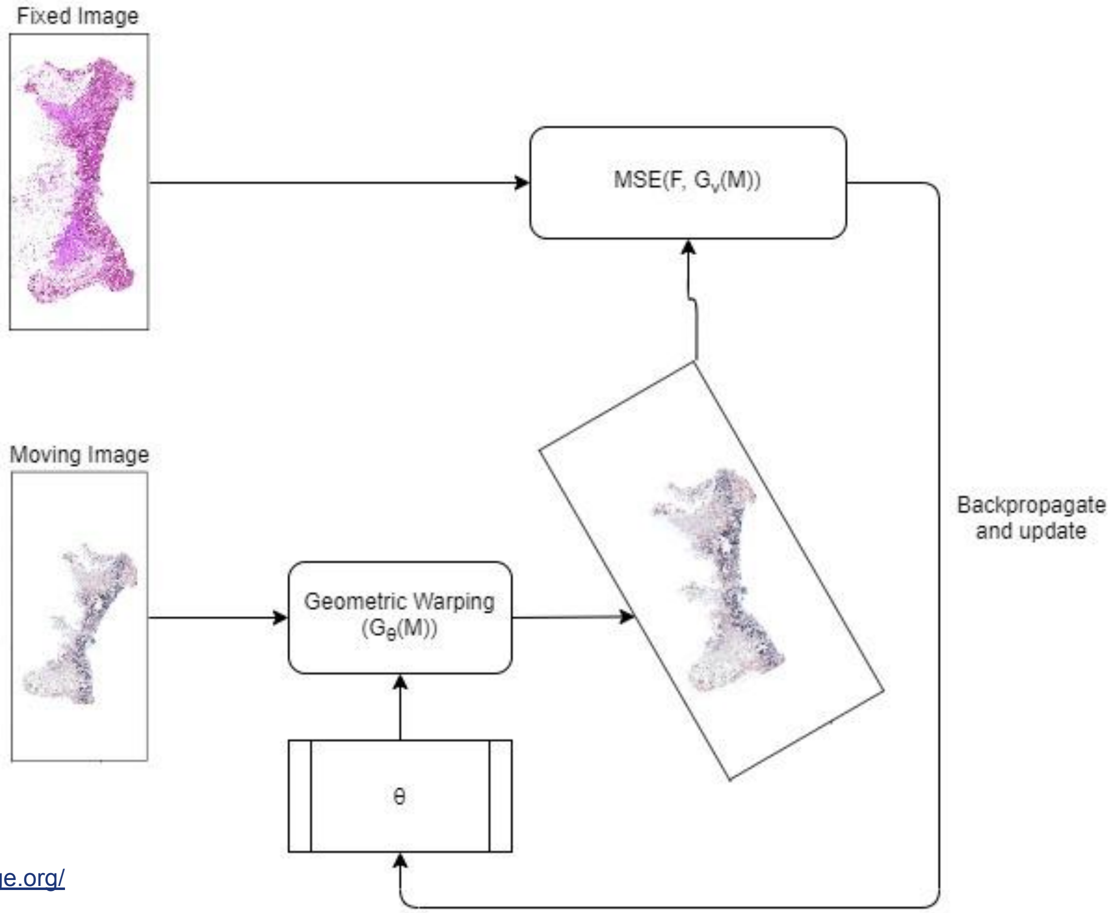
Medical Imaging with Deep Learning, 2020

https://github.com/abnan/DRMIME

Registration



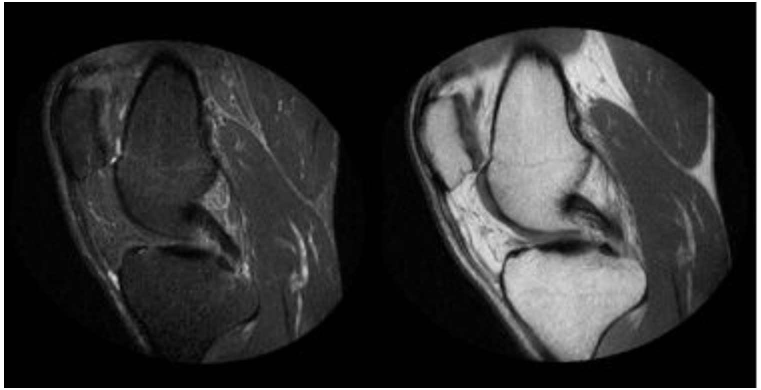
With MSE

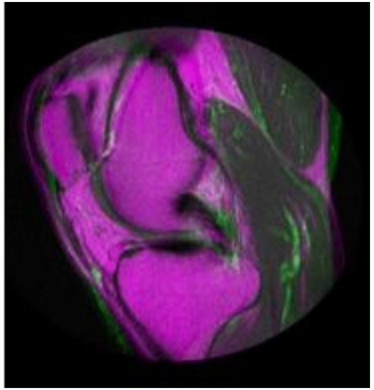


Source: https://anhir.grand-challenge.org/

Problems?

- Multi-modal images
- MSE won't work





Source: https://www.mathworks.com/discovery/image-registration.html

Solution?

- Mutual Information

$$MI(X,Y) = \sum_{x,y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}.$$

Issues?

- Mutual Information for images is computed using joint histograms.
- Histograms are not differentiable.
- No gradient descent?

Differentiable mutual information

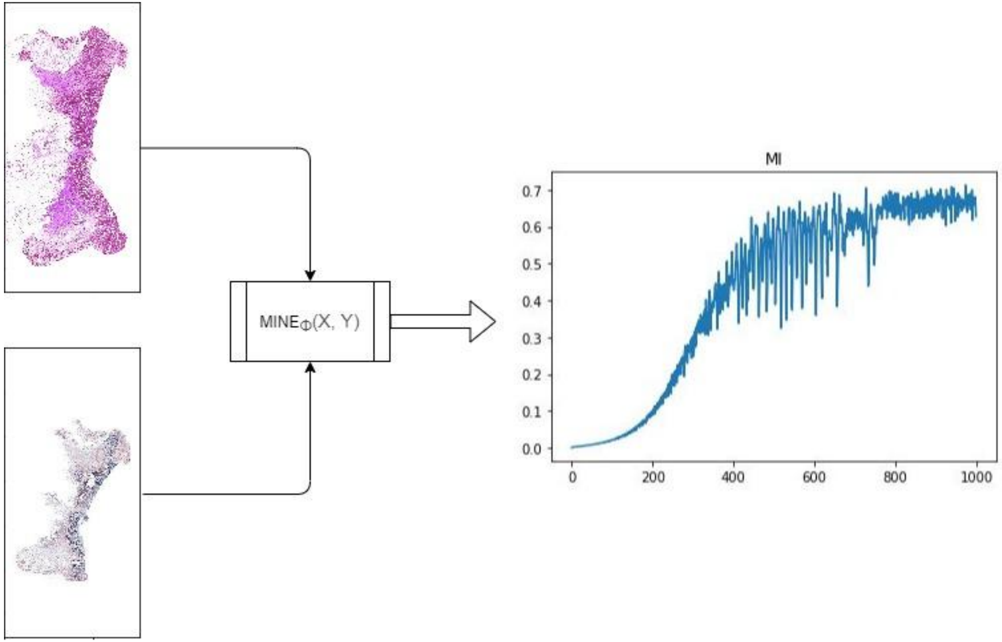
- The function T is realized by a neural network with parameter θ.
- V(θ) is differentiable and can be used as a objective function in place of MI.

Algorithm 1 MINE

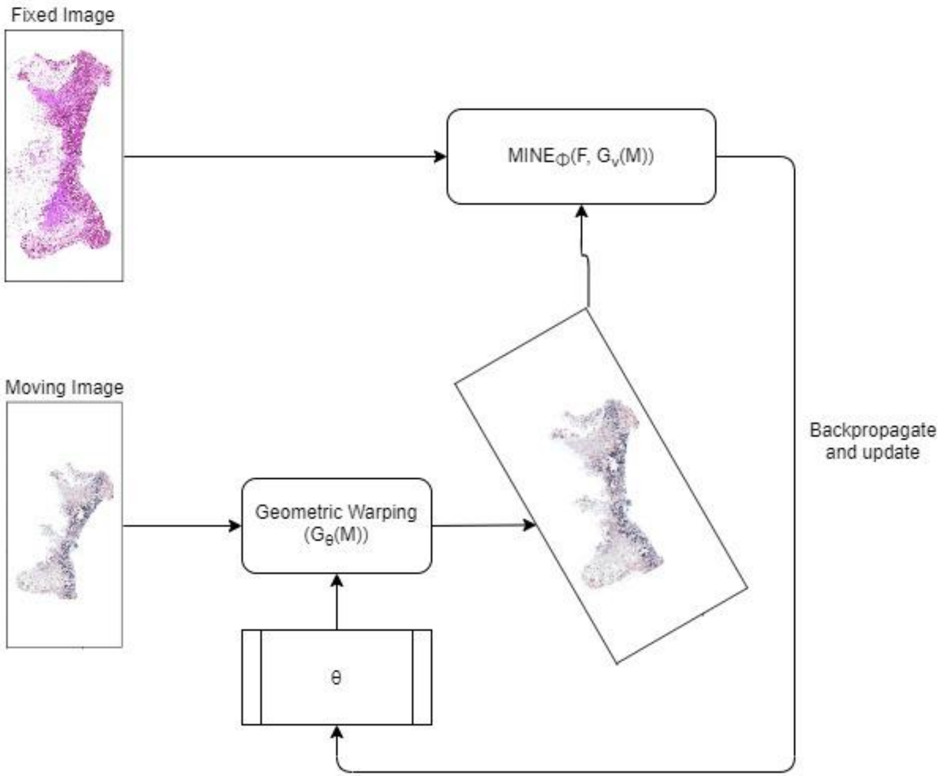
$\theta \leftarrow$ initialize network parameters
repeat
  Draw b minibatch samples from the joint distribution:
  $(\boldsymbol{x}^{(1)}, \boldsymbol{z}^{(1)}), \ldots, (\boldsymbol{x}^{(b)}, \boldsymbol{z}^{(b)}) \sim \mathbb{P}_{XZ}$
  Draw n samples from the Z marginal distribution:
  $\bar{\boldsymbol{z}}^{(1)}, \ldots, \bar{\boldsymbol{z}}^{(b)} \sim \mathbb{P}_Z$
  Evaluate the lower-bound:
  $\mathcal{V}(\theta) \leftarrow \frac{1}{b}\sum_{i=1}^{b} T_\theta(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)}) - \log(\frac{1}{b}\sum_{i=1}^{b} e^{T_\theta(\boldsymbol{x}^{(i)}, \bar{\boldsymbol{z}}^{(i)})})$
  Evaluate bias corrected gradients (e.g., moving average):
  $\widehat{G}(\theta) \leftarrow \widetilde{\nabla}_\theta \mathcal{V}(\theta)$
  Update the statistics network parameters:
  $\theta \leftarrow \theta + \widehat{G}(\theta)$
until convergence

Belghazi, et al. “Mutual information neural estimation,” arXiv:1801.04062v4 [cs.LG] 7 Jun 2018

MINE for images



Currently



Fixed Image
MINEΦ(F, Gv(M))
Moving Image
Geometric Warping
(Gθ(M))
θ
Backpropagate
and update

Matrix exponential

- Matrix exponential of a square matrix A is given by the following:

$$exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!}$$

- Geometric transformation matrices can be obtained by exponential of a linear combination of basis matrices.

Matrix Exponential (Examples)

- Affine transform
  - Basis matrices:

$$B_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, B_3 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B_4 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, B_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  - Transformation matrix:

$$\exp\left(\sum_{i=1}^{6} \theta_i B_i\right) = \sum_{k=0}^{\infty} \frac{\left[\sum_{i=1}^{6} \theta_i B_i\right]^k}{k!}$$

Why matrix exponentials?

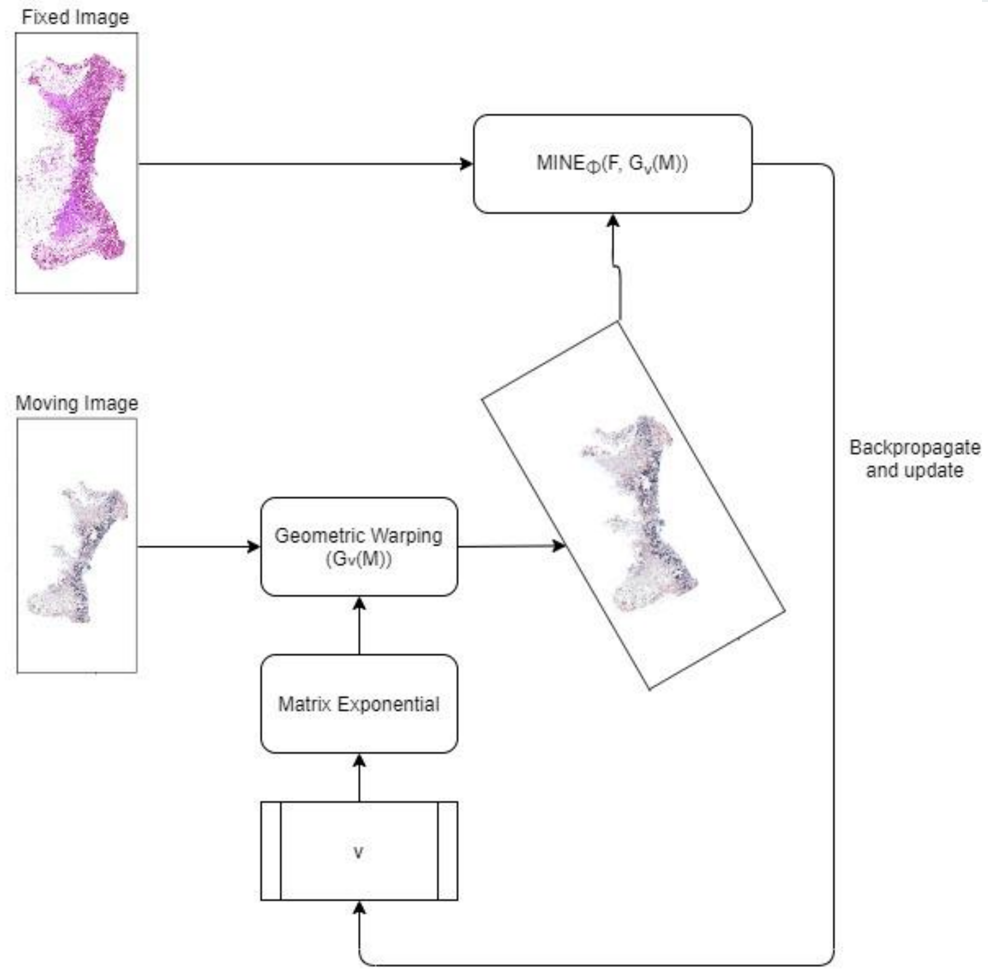
- Consider gradient descent on rotation matrix with these two options:
  - Option 1: Update each element of rotation matrix:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} -= \delta \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

  - Option 2: Update parameter and use matrix exponential (use closed form expression here) when necessary

$$\theta -= \delta\theta$$

So far...



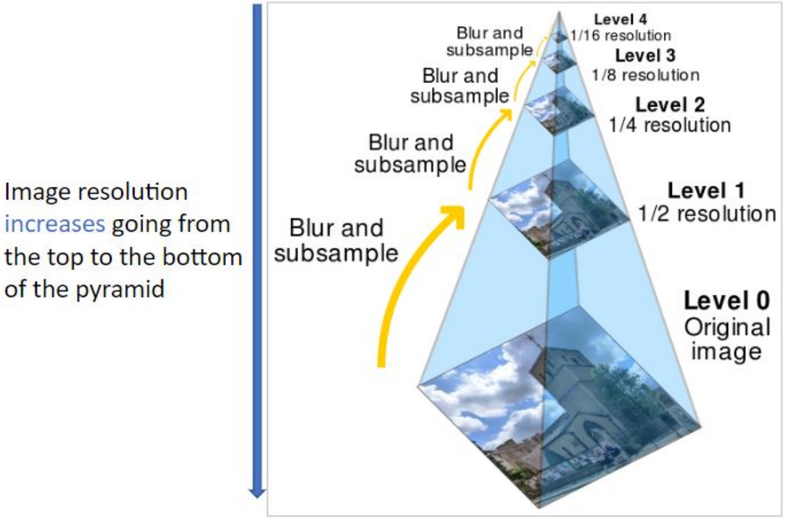
Fixed Image
Moving Image
MINE$_\Phi$(F, G$_v$(M))
Geometric Warping
(Gv(M))
Matrix Exponential
v
Backpropagate
and update

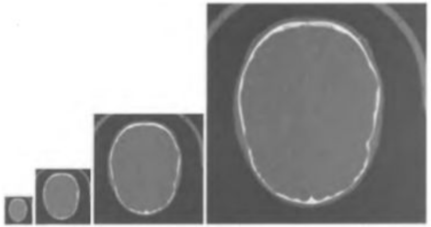
More problems?

- Medical/Microscopy images often are extremely high resolution. So gradient descent can be extremely slow.
- Optimization for neural networks is non-convex.

Solution?

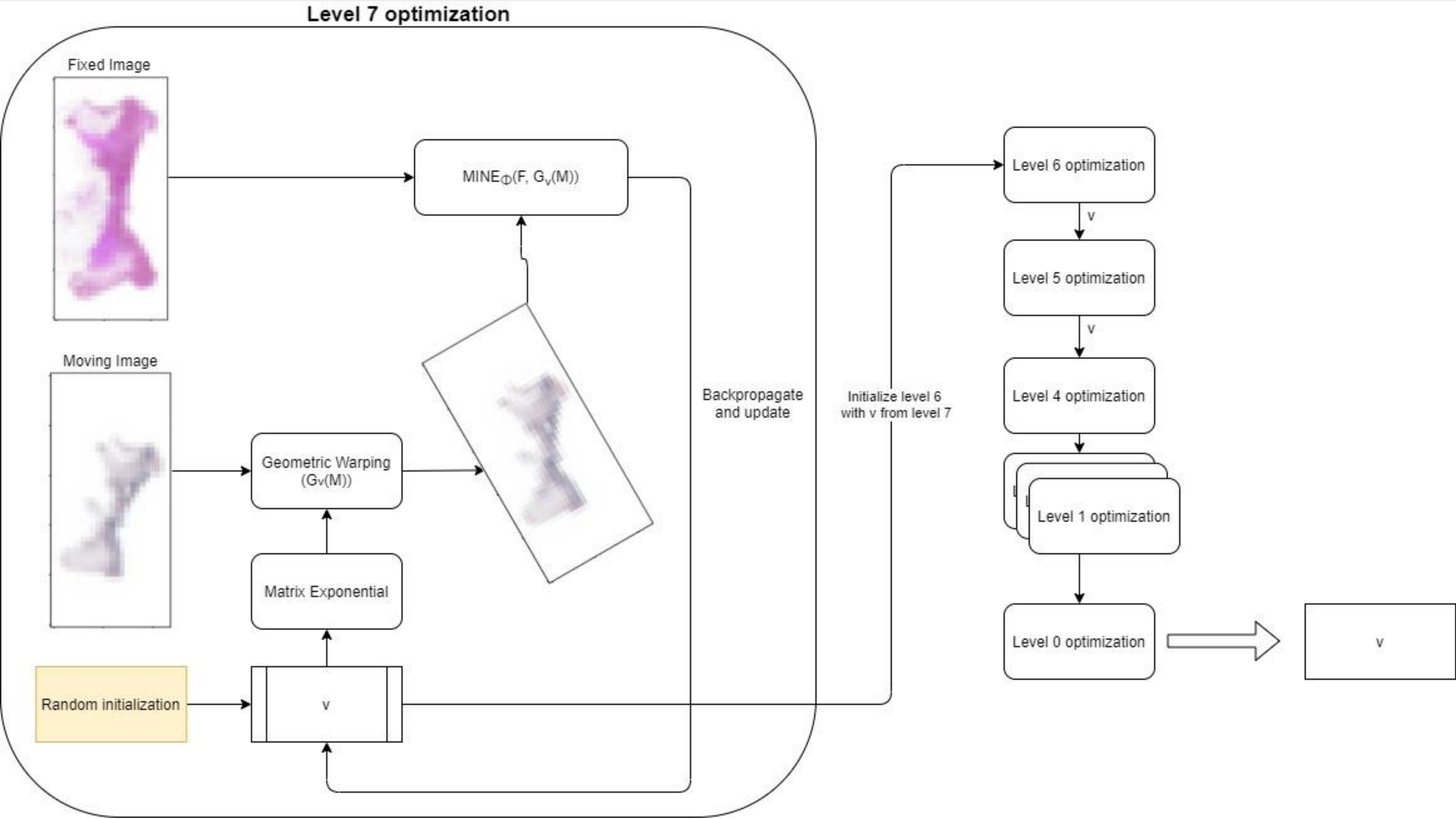
- Gaussian Pyramids



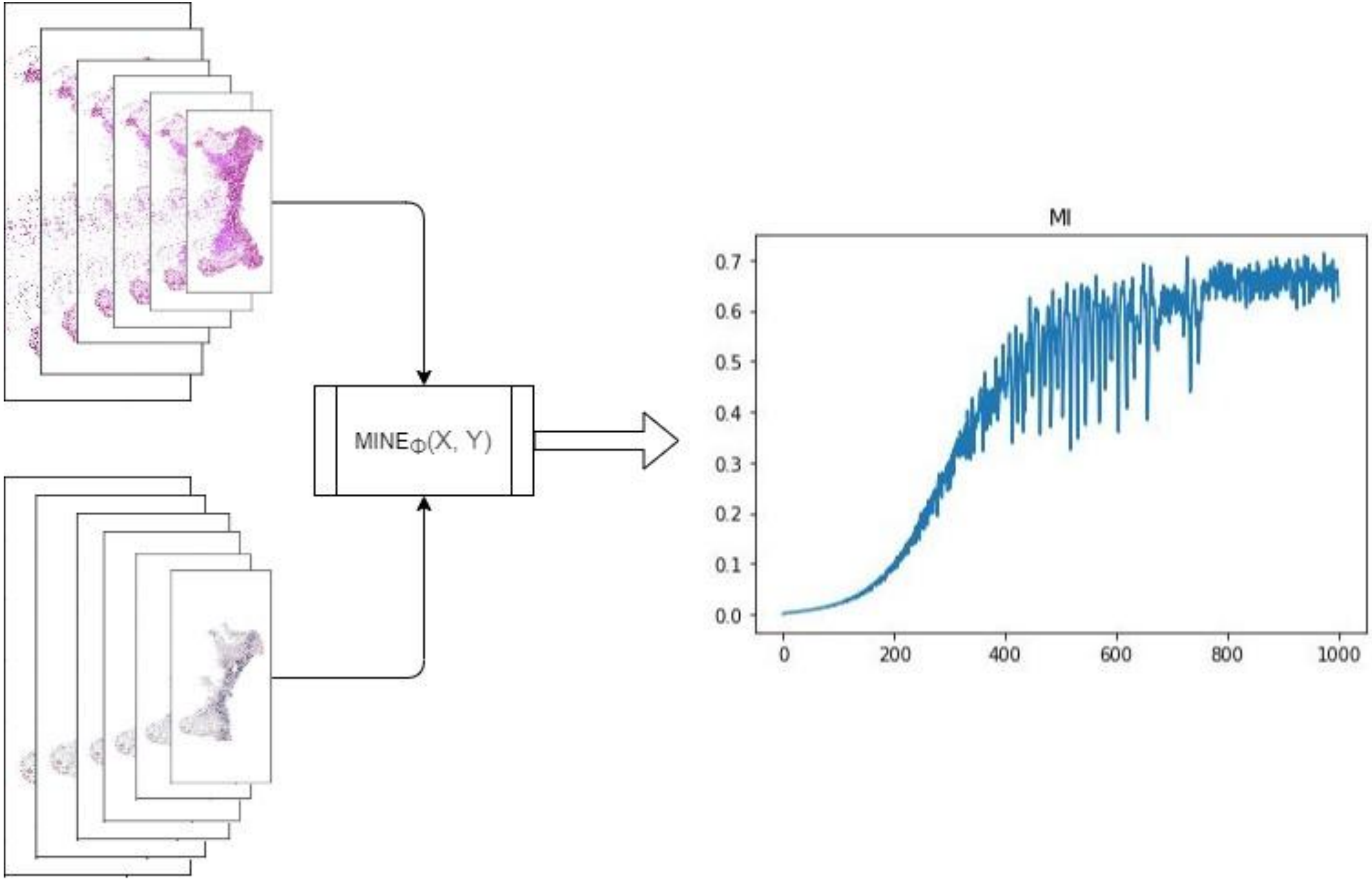


An example image pyramid
Picture source: MICCAI 2010 tutorial

Source: https://en.wikipedia.org/wiki/Pyramid_(image_processing)

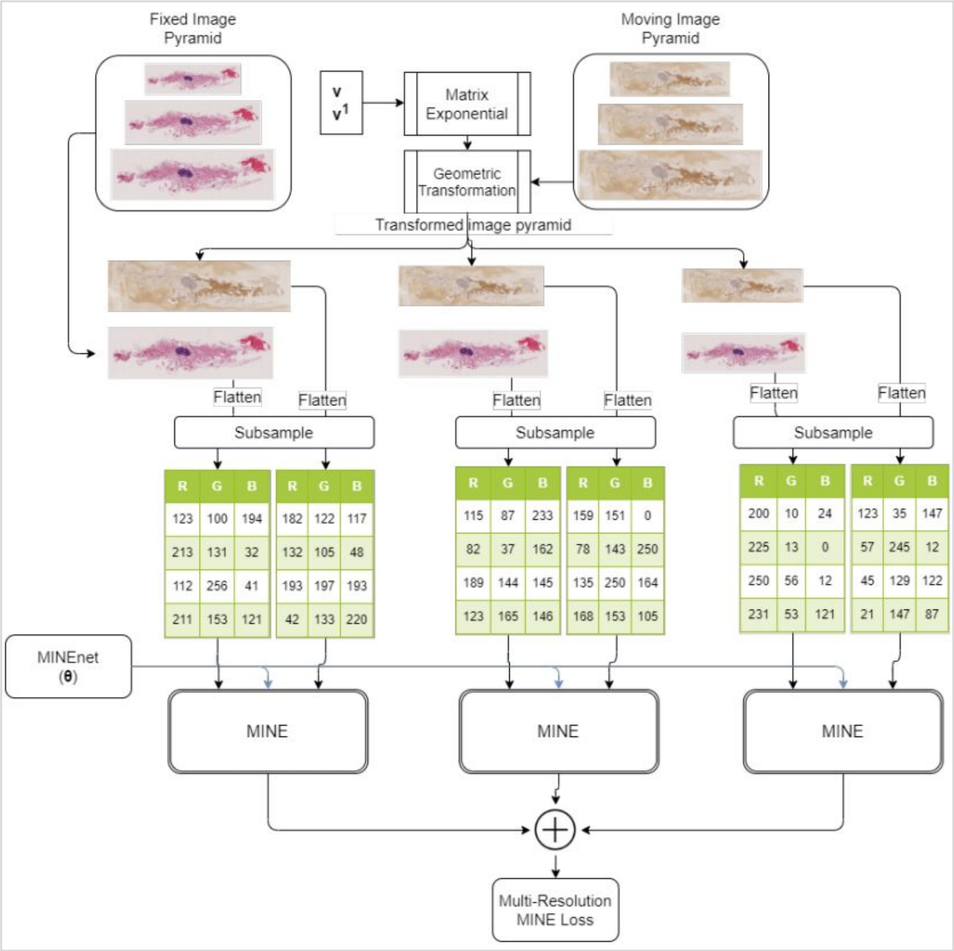

Level 7 optimization
Fixed Image
Moving Image
MINEΦ(F, Gv(M))
Geometric Warping (Gv(M))
Matrix Exponential
Random initialization
v
Backpropagate and update
Initialize level 6 with v from level 7
Level 6 optimization
v
Level 5 optimization
v
Level 4 optimization
Level 1 optimization
Level 0 optimization
v

- Can we do better?
  - What if we did simultaneous optimization for all levels?
- Each level of optimization is for different images. So MI between them changes as well. Solution?
  - A single MINE can be trained for all of these!
- How?
  - Mini-batches can be constructed by sampling from all levels


MINEΦ(X, Y)
MI
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
200
400
600
800
1000

What about the loss?

- Since we are doing simultaneous optimization, with modern deep learning frameworks, it's very easy to combine the loss from each level and perform joint optimization.
- For eg, for just 1 level:
    - Loss = $MI(F, G_v(M))$
- For 4 levels:
    - Loss = $(¼) * [MI(F_1, G_v(M_1)) + MI(F_2, G_v(M_2)) + MI(F_3, G_v(M_3)) + MI(F_4, G_v(M_4))]$


Fixed Image Pyramid
Moving Image Pyramid
v
v1
Matrix Exponential
Geometric Transformation
Transformed image pyramid
Flatten
Flatten
Flatten
Flatten
Flatten
Flatten
Subsample
Subsample
Subsample
R G B
123 100 194
213 131 32
112 256 41
211 153 121
R G B
182 122 117
132 105 48
193 197 193
42 133 220
R G B
115 87 233
82 37 162
189 144 145
123 165 146
R G B
159 151 0
78 143 250
135 250 164
168 153 105
R G B
200 10 24
225 13 0
250 56 12
231 53 121
R G B
123 35 147
57 245 12
45 129 122
21 147 87
MINEnet (θ)
MINE
MINE
MINE
Multi-Resolution MINE Loss

Evaluation

- Public datasets
- Available ground truth

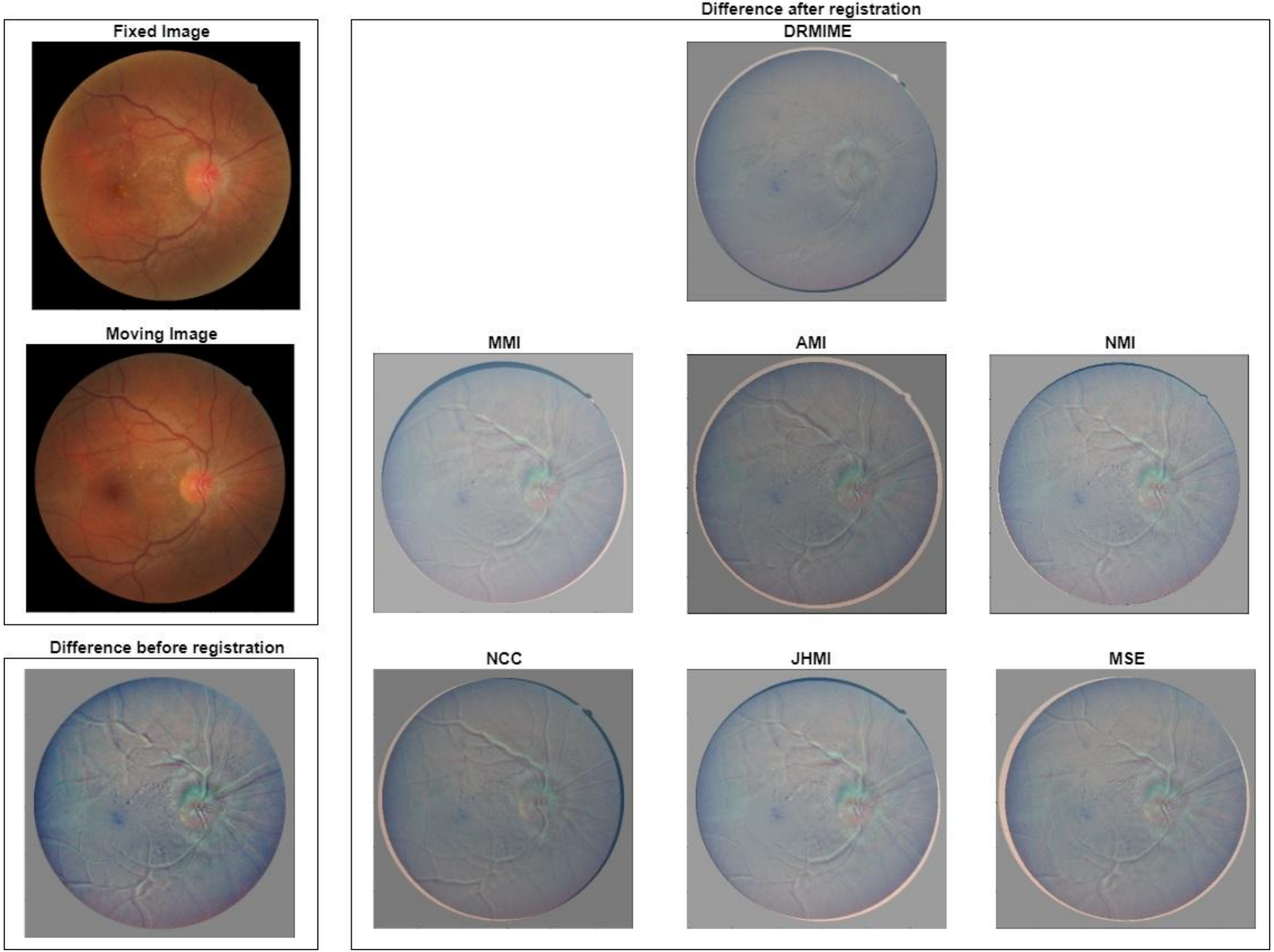
Results

TABLE I: NAED for FIRE dataset along with paired t-test significance values

Algorithm	NAED (Mean ± STD)	p-value
DRMIME	**0.0048** ± 0.014	-
NCC	0.0194 ± 0.033	1.3e-04
MMI	0.0198 ± 0.034	5.4e-05
NMI	0.0228 ± 0.032	1.7e-08
JHMI	0.0311 ± 0.046	4.5e-07
AMI	0.0441 ± 0.028	1.4e-27
MSE	0.0641 ± 0.094	3.5e-03

TABLE II: NAED for ANHIR dataset along with paired t-test significance values

Algorithm	NAED (Mean ± STD)	p-value
DRMIME	**0.0384** ± 0.087	-
NCC	0.0461 ± 0.084	7.0e-04
MMI	0.0490 ± 0.082	6.2e-05
MSE	0.0641 ± 0.094	5.5e-14
NMI	0.0765 ± 0.090	3.0e-31
AMI	0.0769 ± 0.090	3.7e-30
JHMI	0.0827 ± 0.100	8.3e-21



Source: https://projects.ics.forth.gr/cvrl/fire/

Thank you!