Fast Protein Search by Global Shape Similarity Using Krawtchouk Moments as Features

Mindaugas Indriūnas (Mindey)

Interdisciplinary Lifescience Program
Purdue University

February 20, 2012

@Kiharalab

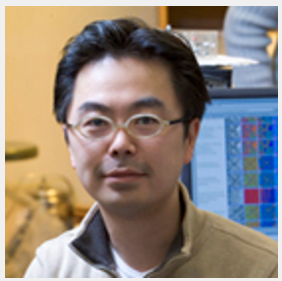
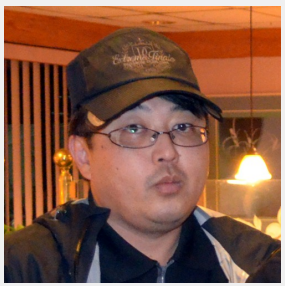
 ,  ,  ,  , 

Figure: Prof. Daisuke Kihara, Dr. Sael Lee, Hyung Rae Kim, Bin Li, David La

Background - 1

- **Problem**: large protein databases are hard to search.
- **79180** of structures in the **Protein Data Bank** *(7 Feb 2012)*
- **2.19 Å** - high resolution of available structures

A solution: Feature Extraction

$$\mathbf{x} = (x^1, x^2, ..., x^{LARGE}) \quad \overset{\mathcal{F}}{\mapsto} (\mathcal{F}(\mathbf{x})^1, \mathcal{F}(\mathbf{x})^2, ..., \mathcal{F}(\mathbf{x})^{SMALL})$$

Figure: Dimensionality-Reduction

Background - 1

- **Problem**: large protein databases are hard to search.
- **79180** of structures in the **Protein Data Bank** *(7 Feb 2012)*
- **2.19 Å** - high resolution of available structures

A solution: Feature Extraction

$$\mathbf{x} = (x^1, x^2, ..., x^{LARGE}) \quad \overset{\mathcal{F}}{\mapsto} (\mathcal{F}(\mathbf{x})^1, \mathcal{F}(\mathbf{x})^2, ..., \mathcal{F}(\mathbf{x})^{SMALL})$$

Figure: Dimensionality-Reduction

Background - 1

- **Problem**: large protein databases are hard to search.
- **79180** of structures in the **Protein Data Bank** *(7 Feb 2012)*
- **2.19 Å** - high resolution of available structures

A solution: Feature Extraction

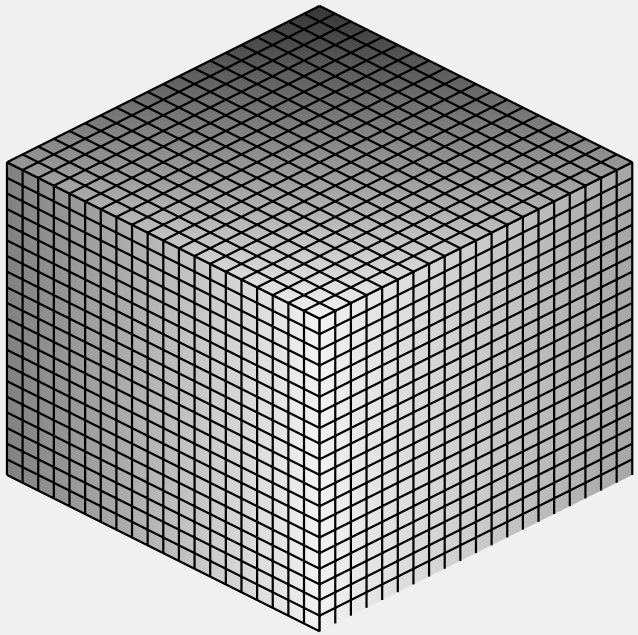
$$\mathbf{x} = (x^1, x^2, ..., x^{LARGE}) \quad \overset{\mathcal{F}}{\mapsto} (\mathcal{F}(\mathbf{x})^1, \mathcal{F}(\mathbf{x})^2, ..., \mathcal{F}(\mathbf{x})^{SMALL})$$

Figure: Dimensionality-Reduction

Background - 1

- **Problem**: large protein databases are hard to search.
- **79180** of structures in the **Protein Data Bank** *(7 Feb 2012)*
- **2.19 Å** - high resolution of available structures

A solution: Feature Extraction



$$\mathbf{x} = (x^1, x^2, ..., x^{LARGE}) \quad \overset{\mathcal{F}}{\mapsto} (\mathcal{F}(\mathbf{x})^1, \mathcal{F}(\mathbf{x})^2, ..., \mathcal{F}(\mathbf{x})^{SMALL})$$

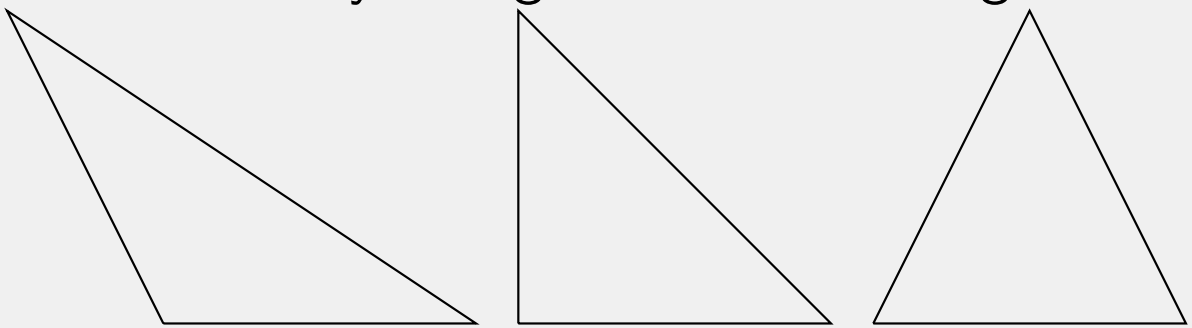
Figure: Dimensionality-Reduction

Background - 2

- **Problem**: large protein databases are hard to search.
- **79180** of structures in the **Protein Data Bank** *(7 Feb 2012)*
- **2.19 Å** - high resolution of available structures

Feature Extraction (Example)

If we want to classify triangles into obtuse, right and acute:



We can speed up the process by creating features by feature map

$$\mathcal{F} : \mathbf{x} = (\alpha, \beta, \gamma) \overset{\mathcal{F}}{\mapsto} (\mathcal{F}(\mathbf{x})^1),$$

Where $\mathcal{F}(\mathbf{x}) = (\max(\mathbf{x}))$.

Figure: Dimensionality-Reduction

Background - 3a

Moments as Features In statistics, the n^{th} raw moment of a discrete distribution is defined as:

$$\mu'_n = \sum_{i=-\infty}^{+\infty} x_i^n \cdot p_i$$

Example: statistical mean is the 1^{st} raw moment:

- 1^{st} raw moment of a sample: $\hat{\mu}'_1 = \sum_{i=1}^{M} x_i \cdot p_i$, where $p_i = \frac{m_i}{N}$ - relative frequency of i^{th} observation in sample ($M \leq N \in \mathbb{N}$).
- $\mu'_1 = (x_1, x_2, ..., x_M) \cdot (p_1, p_2, ..., p_M)^T$ - in terms of matrix notation.
- $\mu'_1 = \langle \vec{x}, \vec{p} \rangle$ - in terms of inner product, "$\vec{x}$ projection on $\vec{p}$"

x - our data, can similarly be projected on other functions, and it is common to denote it as $\langle \mathbf{x}, \cdot \rangle$, where the dot $\cdot$ denotes other function.

Background - 3a

Moments as Features In statistics, the n^{th} raw moment of a discrete distribution is defined as:

$$\mu'_n = \sum_{i=-\infty}^{+\infty} x_i^n \cdot p_i$$

Example: statistical mean is the 1^{st} raw moment:

- 1^{st} raw moment of a sample: $\hat{\mu}'_1 = \sum_{i=1}^{M} x_i \cdot p_i$, where $p_i = \frac{m_i}{N}$ - relative frequency of i^{th} observation in sample ($M \leq N \in \mathbb{N}$).
- $\mu'_1 = (x_1, x_2, ..., x_M) \cdot (p_1, p_2, ..., p_M)^T$ - in terms of matrix notation.
- $\mu'_1 = \langle \vec{x}, \vec{p} \rangle$ - in terms of inner product, "$\vec{x}$ projection on $\vec{p}$"

$\mathbf{x}$ - our data, can similarly be projected on other functions, and it is common to denote it as $\langle \mathbf{x}, \cdot \rangle$, where the dot $\cdot$ denotes other function.

Background - 3a

Moments as Features In statistics, the n^{th} raw moment of a discrete distribution is defined as:

$$\mu'_n = \sum_{i=-\infty}^{+\infty} x_i^n \cdot p_i$$

Example: statistical mean is the 1^{st} raw moment:

- 1^{st} raw moment of a sample: $\hat{\mu}'_1 = \sum_{i=1}^{M} x_i \cdot p_i$, where $p_i = \frac{m_i}{N}$ - relative frequency of i^{th} observation in sample ($M \leq N \in \mathbb{N}$).
- $\mu'_1 = (x_1, x_2, ..., x_M) \cdot (p_1, p_2, ..., p_M)^T$ - in terms of matrix notation.
- $\mu'_1 = \langle \vec{x}, \vec{p} \rangle$ - in terms of inner product, "$\vec{x}$ projection on $\vec{p}$"

$\mathbf{x}$ - our data, can similarly be projected on other functions, and it is common to denote it as $\langle \mathbf{x}, \cdot \rangle$, where the dot $\cdot$ denotes other function.

Background - 3a

Moments as Features In statistics, the n^{th} raw moment of a discrete distribution is defined as:

$$\mu'_n = \sum_{i=-\infty}^{+\infty} x_i^n \cdot p_i$$

Example: statistical mean is the 1^{st} raw moment:

- 1^{st} raw moment of a sample: $\hat{\mu}'_1 = \sum_{i=1}^{M} x_i \cdot p_i$, where $p_i = \frac{m_i}{N}$ - relative frequency of i^{th} observation in sample ($M \leq N \in \mathbb{N}$).
- $\mu'_1 = (x_1, x_2, ..., x_M) \cdot (p_1, p_2, ..., p_M)^T$ - in terms of matrix notation.
- $\mu'_1 = \langle \vec{x}, \vec{p} \rangle$ - in terms of inner product, "$\vec{x}$ projection on $\vec{p}$"

$\mathbf{x}$ - our data, can similarly be projected on other functions, and it is common to denote it as $\langle \mathbf{x}, \cdot \rangle$, where the dot $\cdot$ denotes other function.

Background - 3a

Moments as Features In statistics, the n^{th} raw moment of a discrete distribution is defined as:

$$\mu'_n = \sum_{i=-\infty}^{+\infty} x_i^n \cdot p_i$$

Example: statistical mean is the 1^{st} raw moment:

- 1^{st} raw moment of a sample: $\hat{\mu}'_1 = \sum_{i=1}^{M} x_i \cdot p_i$, where $p_i = \frac{m_i}{N}$ - relative frequency of i^{th} observation in sample ($M \leq N \in \mathbb{N}$).
- $\mu'_1 = (x_1, x_2, ..., x_M) \cdot (p_1, p_2, ..., p_M)^T$ - in terms of matrix notation.
- $\mu'_1 = \langle \vec{x}, \vec{p} \rangle$ - in terms of inner product, "$\vec{x}$ projection on $\vec{p}$"

x - our data, can similarly be projected on other functions, and it is common to denote it as $\langle \mathbf{x}, \cdot \rangle$, where the dot $\cdot$ denotes other function.

Background - 3b

Other commonly known moments in **inner product notation:**

Variance

$$\mu_2 = E\left[(X-\mu)^2\right] = \left\langle (X-\mu)^2, \vec{p}_{X-\mu} \right\rangle$$

Skewness

$$\mu_3 = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^3, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Kurtosis

$$\mu_4 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^4, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Main idea: moments characterize shape of a distribution.

More generally - shape of a function, such as $f(x, y, z)$ - our data.

Background - 3b

Other commonly known moments in **inner product notation:**

Variance

$$\mu_2 = E\left[(X-\mu)^2\right] = \left\langle (X-\mu)^2, \vec{p}_{X-\mu} \right\rangle$$

Skewness

$$\mu_3 = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^3, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Kurtosis

$$\mu_4 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^4, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Main idea: moments characterize shape of a distribution.

More generally - shape of a function, such as $f(x, y, z)$ - our data.

Background - 3b

Other commonly known moments in **inner product notation:**

Variance

$$\mu_2 = E\left[(X-\mu)^2\right] = \left\langle (X-\mu)^2, \vec{p}_{X-\mu} \right\rangle$$

Skewness

$$\mu_3 = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^3, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Kurtosis

$$\mu_4 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^4, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Main idea: moments characterize shape of a distribution.

More generally - shape of a function, such as $f(x, y, z)$ - our data.

Background - 3b

Other commonly known moments in **inner product notation:**

Variance

$$\mu_2 = E\left[(X-\mu)^2\right] = \left\langle (X-\mu)^2, \vec{p}_{X-\mu} \right\rangle$$

Skewness

$$\mu_3 = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^3, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Kurtosis

$$\mu_4 = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \left\langle \left(\frac{X-\mu}{\sigma}\right)^4, \vec{p}_{\frac{X-\mu}{\sigma}} \right\rangle$$

Main idea: moments characterize shape of a distribution.
More generally - shape of a function, such as $f(x, y, z)$ - our data.

Background - 3c (Dr. Sael Lee's focus)

3D Zernike Descriptors - data f projections in Zernike functions:
$Z_{nl}^m(r,\varphi,\phi) = R_{nl}(r)Y_l^m(\varphi,\phi)$, where $Y_l^m(\varphi,\phi)$ are spherical harmonics of the l^{th} degree with $l \leq n,\ m \in [-l,l]$, and $n-l$ even non-negative.
$R_{nl}(r)$ are radial polynomials, where r is the radius defined so that $Z_{nl}^m(x)$ are orthonormal polynomials, when written in Cartesian coordinates.
For a 3D function $f(x)$ where $x \in \mathbb{R}^3$, the 3D Zernike moments are given by:

$$\Omega_{nl}^m = \langle f, Z_{nl}^m \rangle = \frac{3}{4\pi} \sum_{r+s+t \leq n} \chi_{nlm}^{rst} M_{rst}$$

The above equation is expressed as a linear combination of geometric moments of order n where M_{rst} denotes the geometrical moment of the object normalized to fit in the unit sphere and χ_{nlm}^{rst} is a set of complex coefficients.
Since thus defined moments are not rotationally invariant, they are collected into $(2l+1)$-dimensional vectors and the norms $(||\cdot||)$ of the vectors
$\Omega_{nl} = \left[\Omega_{nl}^l, ..., \Omega_{nl}^{-l}\right]$ define the rotationally invariant 3D Zernike descriptors:

$$F_{nl} = ||\Omega_{nl}||$$

$$\mathcal{F}: \ \left(x^1, x^2, ..., x^{LARGE}\right) \mapsto \left[F_{nl}^1(\mathbf{x}), F_{nl}^2(\mathbf{x}), , ..., F_{nl}^{SMALL}(\mathbf{x})\right]$$

"Hypothesis" - 1 (My focus)

3D Krawtchouk Descriptors - projections in Krawtchouk functions:

Weighted Krawtchouk Polynomials

$\overline{K}(x;p,N) = K_n(x;p,N)\sqrt{\frac{w(x;p,N)}{\rho(n;p,N)}}$ on $x, n = 0, 1, 2...N,\ N > 0,\ p \in (0,1)$

$K_n(x;p,N) =_2 F_1(-n,-x;-N;\frac{1}{2})$, - non-weighted Krawtchouk polynomials,
$w(x;p,N) = \binom{N}{x} p^x (1-p)^{N-x}$ - weight function,
$\rho(n;p,N) = (-1)^n \left(\frac{1-p}{p}\right)^n \frac{n!}{(-N)_n}$ - normalization constant.

Hypergeometric function

${}_2F_1(a,b;c;z) = \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{(c)_k} \frac{z^k}{k!}$, where $(a)_k = a(a+1)(a+2)...(a+k-1)$.

The projections of f on $\overline{K}(x;p,N)$:

$$\langle f, \overline{K} \rangle$$

- are called Weighted Krawtchouk Moments of function f.
The moments in 3D case were introduced by (Mademlis et.al.)

"Hypothesis" - 1 (My focus)

3D Krawtchouk Descriptors - projections in Krawtchouk functions:

Weighted Krawtchouk Polynomials

$\overline{K}(x;p,N) = K_n(x;p,N)\sqrt{\frac{w(x;p,N)}{\rho(n;p,N)}}$ on $x, n = 0, 1, 2...N,\ N > 0,\ p \in (0,1)$

$K_n(x;p,N) =_2 F_1(-n,-x;-N;\frac{1}{2})$, - non-weighted Krawtchouk polynomials,
$w(x;p,N) = \binom{N}{x} p^x (1-p)^{N-x}$ - weight function,
$\rho(n;p,N) = (-1)^n \left(\frac{1-p}{p}\right)^n \frac{n!}{(-N)_n}$ - normalization constant.

Hypergeometric function

${}_2F_1(a,b;c;z) = \sum_{k=0}^{\infty} \frac{(a)_k (b)_k}{(c)_k} \frac{z^k}{k!}$, where $(a)_k = a(a+1)(a+2)...(a+k-1)$.

The projections of f on $\overline{K}(x;p,N)$:

$$\langle f, \overline{K} \rangle$$

- are called Weighted Krawtchouk Moments of function f.
The moments in 3D case were introduced by (Mademlis et.al.)

"Hypothesis" - 2 (My focus)

Weighted 3D Krawtchouk Moments (Mademlis et.al. 2006)

Given $f(x, y, z)$ - a 3D function defined in a discrete field
$A = (x, y, z) : \ x, y, z \in \mathbb{N}, \ x = [0...N-1], y = [0...M-1], z = [0...L-1]$
Weighted 3D Krawtchouk Moments of order $(n + m + l)$ of f:

$$\overline{Q}_{nml} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \sum_{z=0}^{L-1} \overline{K}_n(x; p_x, N-1) \times \overline{K}_m(y; p_y, M-1) \times \overline{K}_l(z; p_z, L-1) \times f(x, y, z)$$

Weighted Krawtchouk moments can be used as descriptor of any 3D object, if it can be expressed as a function $f(x, y, z)$ defined in a discrete space $[0...N-1] \ \times \ [0...M-1] \ \times \ [0...L-1]$, e.g., if model is expressed as a binary volumetric function (e.g., 3D grid with voxels each carrying 1 bit of information).
The descriptor vector then is defined as:
$D = \left[\overline{Q}_{nml} | n + m + l \in [0...s]\right]$
Notice that feature map:

$$\mathcal{F} : \ \left(x^1, x^2, ..., x^{LARGE}\right) \mapsto \left[\overline{Q}^1_{nml}(\mathbf{x}), \overline{Q}^2_{nml}(\mathbf{x}), , ..., \overline{Q}^{SMALL}_{nml}(\mathbf{x})\right]$$

Goals

- Implement the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- Compute 3D Krawtchouk descriptors for a set of proteins.
- Create a simple classifier working based on the features.
- Evaluate classification and retrieval performance.
- Compare it to retrieval with other available methods, like TM-Align.

Goals

- Implement the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- Compute 3D Krawtchouk descriptors for a set of proteins.
- Create a simple classifier working based on the features.
- Evaluate classification and retrieval performance.
- Compare it to retrieval with other available methods, like TM-Align.

Goals

- Implement the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- Compute 3D Krawtchouk descriptors for a set of proteins.
- Create a simple classifier working based on the features.
- Evaluate classification and retrieval performance.
- Compare it to retrieval with other available methods, like TM-Align.

Goals

- Implement the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- Compute 3D Krawtchouk descriptors for a set of proteins.
- Create a simple classifier working based on the features.
- Evaluate classification and retrieval performance.
- Compare it to retrieval with other available methods, like TM-Align.

Goals

- Implement the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- Compute 3D Krawtchouk descriptors for a set of proteins.
- Create a simple classifier working based on the features.
- Evaluate classification and retrieval performance.
- Compare it to retrieval with other available methods, like TM-Align.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - PyLaya
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - PyLibSVM
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - PyMVPA
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - 30,717 x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - 30,717 x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - 30,717 x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - 30,717 x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - 30,717 x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - 30,717 x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 1 - (Tools, Computational Environment)

Software

- Data standardization and PCA -
  - **Octave** (Opensource MATLAB equivalent)
- Computation of $\overline{K}$ matrix -
  - **Maple** (Maplesoft, proprietary, licensed)
  - Custom **C** code
- Classifier and ROC analysis -
  - **Python**
    - NumPy
    - Pytave
  - Custom **C** code

Hardware

- **Kiharalab** computers
- **BoilerGrid** computers
  - **30,717** x86_64-Linux cores using Condor system.

Methods - 2 (Technical Details)

Computing the $\overline{K}(x; p, N)$ - Weighted Krawtchouk Polynomials

Maple code

```
> restart:  with(plots):
> K := (n,x,p,N) -> hypergeom([-n,-x],[-N],1/p);
```

$$K := (n,\ x,\ p,\ N) \to \mathrm{hypergeom}([-n,\ -x],\ [-N],\ \frac{1}{p})$$

$$\omega := (x,\ p,\ N) \to \mathrm{binomial}(N,\ x)\, p^{x}\, (1-p)^{(N-x)}$$

$$\rho := (n,\ p,\ N) \to \frac{(-1)^{n}\, (\frac{1-p}{p})^{n}\, n!}{\mathrm{pochhammer}(-N,\ n)}$$

```
> Kw := (n,x,p,N) -> K(n,x,p,N)*sqrt(omega(x,p,N)/rho(n,p,N));
```

$$Kw := (n,\ x,\ p,\ N) \to \mathrm{K}(n,\ x,\ p,\ N)\, \sqrt{\frac{\omega(x,\ p,\ N)}{\rho(n,\ p,\ N)}}$$

```
> Kw(1,3,0.5,10);  We get Weighted Krawtchouk Polynomials
```

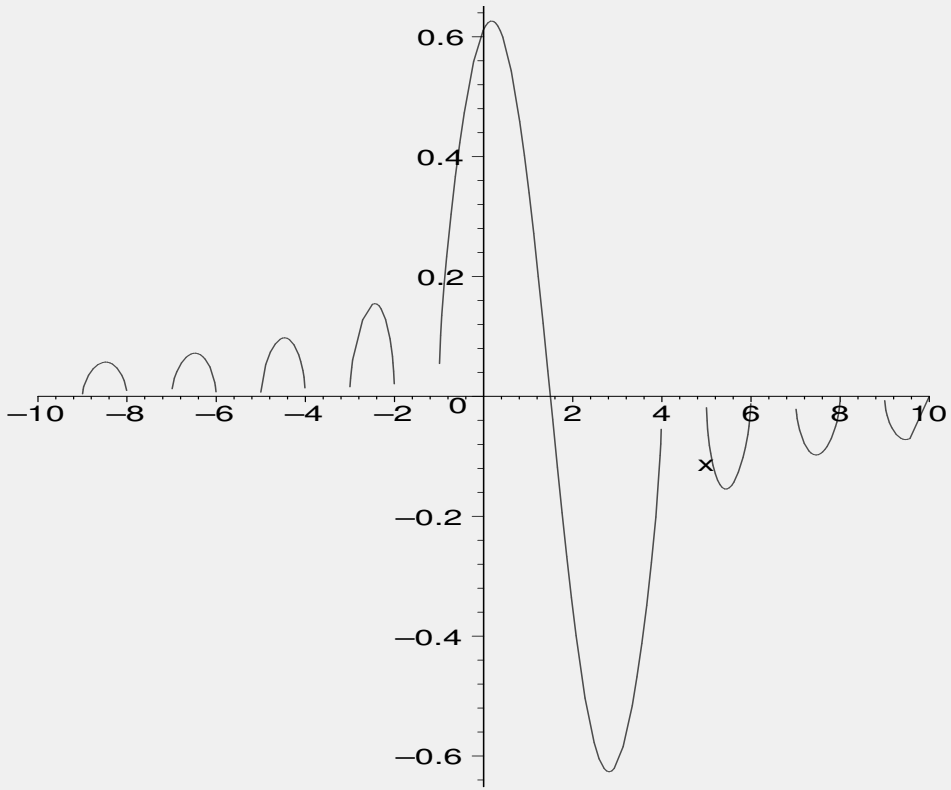
0.4330127020

```
> plot(Kw(1,x,0.5,3),x);  n = 1, p = 0.5, N = 3
```

Methods - 2 (Technical Details)

Plot of a weighted Krawtchouk polynomial

```
> plot(Kw(1,x,0.5,3),x);
```

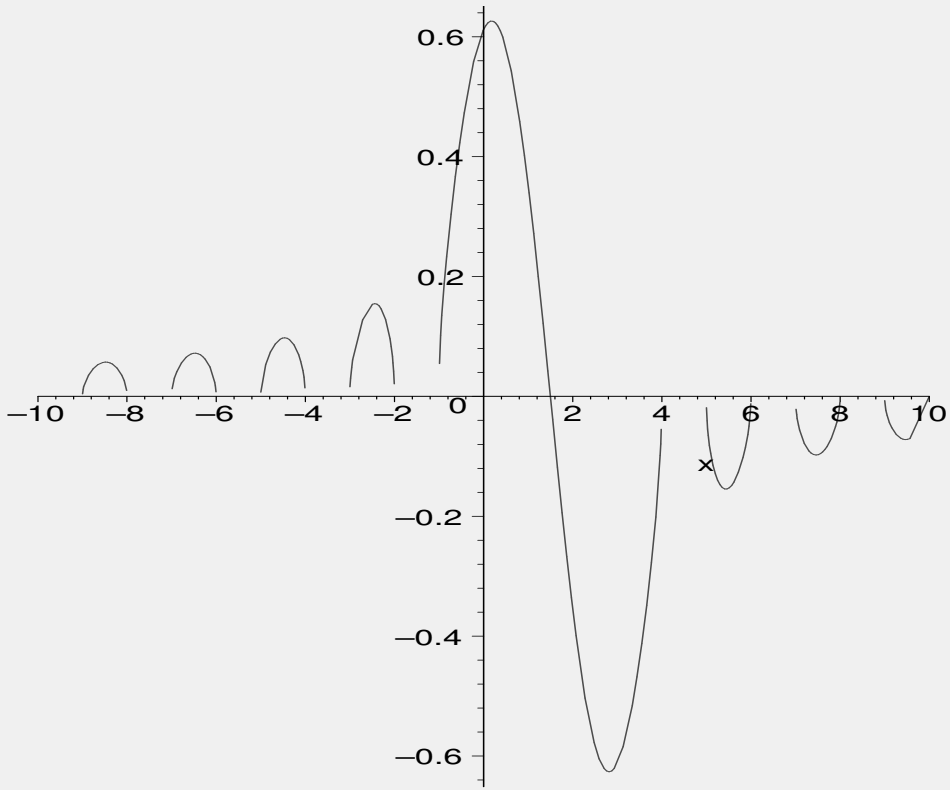


- Obtain and save $\overline{K}(n, x, p, N)$ for $n = [0..159]$, $x = [0..159]$, with $p = 0.5$, $N = 160$.

Methods - 2 (Technical Details)

Plot of a weighted Krawtchouk polynomial

```
> plot(Kw(1,x,0.5,3),x);
```



- Obtain and save $\overline{K}(n, x, p, N)$ for $n = [0..159],\ x = [0..159]$, with $p = 0.5,\ N = 160$.

Methods - 3 (Technical Details)

Weighted 3D Krawtchouk Moments of order $(n+m+l)$ of f

$$\overline{Q}_{nml} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \sum_{z=0}^{L-1} \overline{K}_n(x; p, N-1) \overline{K}_m(y; p, M-1) \overline{K}_l(z; p, L-1) f(x, y, z)$$

Quick multiplication with C

```
FILE *poly = fopen( argv[1], "r" ); # file containing precomputed values of Krawtchouk Polynomials
FILE *file = fopen( argv[2], "r" ); # file containing protein 3D grid data

// --- START --- //

for (i = 0; i < size_poly; i++) fscanf(poly, "%f", &info[i]);
for (i = 0; i < size_file; i++) fscanf(file, "%f", &data[i]);

for (x = 0; x <= 159; x++)
    for (y = 0; y <= 159; y++)
        for (z = 0; z <= 159; z++)
             s += info[n*160+x]*info[m*160+y]*info[l*160+z]*data[x*25600+y*160+z];

printf("%f\n", s);

// --- FINISH --- //

free(info); free(data);
```

Methods - 3 (Technical Details)

Weighted 3D Krawtchouk Moments of order $(n+m+l)$ of f

$$\overline{Q}_{nml} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} \sum_{z=0}^{L-1} \overline{K}_n(x; p, N-1) \overline{K}_m(y; p, M-1) \overline{K}_l(z; p, L-1) f(x, y, z)$$

Quick multiplication with C

```
FILE *poly = fopen( argv[1], "r" ); # file containing precomputed values of Krawtchouk Polynomials
FILE *file = fopen( argv[2], "r" ); # file containing protein 3D grid data

// --- START --- //

for (i = 0; i < size_poly; i++) fscanf(poly, "%f", &info[i]);
for (i = 0; i < size_file; i++) fscanf(file, "%f", &data[i]);

for (x = 0; x <= 159; x++)
    for (y = 0; y <= 159; y++)
        for (z = 0; z <= 159; z++)
             s += info[n*160+x]*info[m*160+y]*info[l*160+z]*data[x*25600+y*160+z];

printf("%f\n", s);

// --- FINISH --- //

free(info); free(data);
```

Methods - 3 (Technical Details)

Octave code - for data prep. and PCA

```
function [Y] = fun(fname)
D = load (fname, '-ascii');
query = reshape(D, 160, 160, 160);
X = [];
for i=1:160,
    for j=1:160,
        for k=1:160,
            if query(i,j,k) == 1,
                X = [X; [i j k]];
            end
        end
    end
end
EX = mean(X); CX = X - repmat(EX,rows(X),1);
p = princomp(CX);
P = [];
P(:,:,1) = [p(1:end,1) p(1:end,2) -p(1:end,3)];
  ...
for i=1:8,
    x_result = 80+round(CX*P(:,:,i));
    x_output = zeros(160, 160, 160);
    for m=1:rows(x_result),
        x_output(x_result(m,1),x_result(m,2),x_result(m,3)) = 1;
    end
    x_data = reshape(x_output, 1, 4096000);
    file_name = strcat('/home/mindey/kihara/glued160/example/x_',num2str(i));
    x_fid = fopen(file_name, 'w');
    fprintf(x_fid, ' %i', x_data);
    fclose(x_fid);
end
```

Methods - 3 (Technical Details)

Python code - 1

```
# 1. Read .grid filenames in the folder
import os
import pytave
import numpy
import shlex, subprocess

dir = '/home/mindey/kihara/glued160/example'
filenames = os.listdir(dir)
filelist = []
for filename in filenames:
    if ('.grid' in filename):
        filelist.append(filename)

# 2. For each file do both normal PCA, and 8 types of pca fo each protein in the database, and compute
# their moments. Save these moments to files.
for fn in filelist:
    loc = dir+'/'+str(fn)
    fm = fn[0:-4]+'8moms'
    if fm not in filenames:
        F = open(dir+'/'+fm, 'w')
        # Doing PCA, writing x_1, x_2, ..., x_8
        res = pytave.feval(1, "fun", loc)
        for it in range(8):
            if it < 8:
                eol = '\n'
            # Computing moments
            m = os.popen("./sum M.txt /home/mindey/kihara/glued160/example/x_"+str(it+1))
            M = m.readlines()[0]+eol
            # Writing them to files with .8moms extension
            F.write(M)
```

Methods - 3 (Technical Details)

Python code - 2

```
            # Writing them to files with .8moms extension
            F.write(M)
        os.system("rm /home/mindey/kihara/glued160/example/x_*")
        F.close()
        #M = numpy.array([float(s) for s in m[0].split()])

# Getting filenames again
filenames = os.listdir(dir)
filelist = []
counter = 0
for ix, filename in enumerate(filenames):
    if ('.8moms' in filename):
        filelist.append(filename)
        print str(counter)+". ", filelist[counter]
        counter += 1

# Letting the user choose query
query_id = int(raw_input('Choose query id: '))
# Reading the query protein's eighth line
pcaln = open(dir+'/'+filelist[query_id], 'r')
# print dir+'/'+filelist[query]
query = numpy.array([float(val) for val in pcaln.readlines()[7].split()])
pcaln.close()

# Defining distance
distance = lambda x, y: (sum(abs(x-y)**2))**0.5
# Creating database of remaining moments
database = []
# By opening each other protein's 8mom file and choose the descriptor which is nearest to the query
```

Methods - 3 (Technical Details)

Python code - 3

```
# By opening each other protein's 8mom file and choose the descriptor which is nearest to the query
for ix, filename in enumerate(filelist):
    m8 = open(dir+'/'+filename, 'r').readlines()
    d8 = []
    for line in m8:
        pca8 = numpy.array([float(val) for val in line.split()])
        d8.append(distance(query, pca8))
    # Search for nearest moments to the "pca" by distance, and append it to database
    D8 = numpy.array(d8)
    # With normal PCA alignment:
    # database.append(numpy.array([float(val) for val in m8[7].split()]))
    # With optimized (argmin) PCA alignment:
    database.append(numpy.array([float(val) for val in m8[int(D8.argmin())].split()]))

# Query database for distances to query
distances = {}
for db_id, descriptors in enumerate(database):
    distances[db_id] = distance(query, descriptors)

# Sort the dictionary of results
order = sort_by_values(distances)

# Print query results
print 'Query: '+str(query_id)+'. '+filelist[query_id]
print 'Results: '
for ix in order:
    if ix < 10:
        print ' '+str(ix)+'.',
    else:
        print str(ix)+'.',
```

Methods - 3 (Technical Details)

I had also used some code for running programs in Condor, by an example provided by PhD. student David La.

Example code run on Condor

```
# Header Stuff

Executable = test.py
Universe = vanilla
notification=never

requirements = ( ( OpSys == "LINUX" ) && ( regexp("hamlet",Name) == FALSE ) && ( machine != "dragon.bio.purd

should_transfer_files = IF_NEEDED
when_to_transfer_output = ON_EXIT

on_exit_remove = ( (ExitBySignal == False) && (ExitCode == 0) )

# First Run
Arguments =
Output = ./output/test1_$(process).out
Error = ./err/test1_.$(process).err
LOG = ./log/test1_.$(process).log
transfer_input_files =  ./pdbs/1bi9-A.pdb, ./pdbs/1moh.pdb, CE, ./pom/mkDB, ./pom/mkDB_sgi, ./pom/mkDB_sun,
Queue
```

Conclusions - 1 (Results)

- **Implemented** the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- **Computed** 3D Krawtchouk descriptors for a set of proteins.
- **Created** a simple classifier working based on the features.
- **Evaluated** classification and retrieval performance.
- **Compared** it to retrieval with other available method: TM-Align.

Conclusions - 1 (Results)

- **Implemented** the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- **Computed** 3D Krawtchouk descriptors for a set of proteins.
- **Created** a simple classifier working based on the features.
- **Evaluated** classification and retrieval performance.
- **Compared** it to retrieval with other available method: TM-Align.

Conclusions - 1 (Results)

- **Implemented** the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- **Computed** 3D Krawtchouk descriptors for a set of proteins.
- **Created** a simple classifier working based on the features.
- **Evaluated** classification and retrieval performance.
- **Compared** it to retrieval with other available method: TM-Align.

Conclusions - 1 (Results)

- **Implemented** the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- **Computed** 3D Krawtchouk descriptors for a set of proteins.
- **Created** a simple classifier working based on the features.
- **Evaluated** classification and retrieval performance.
- **Compared** it to retrieval with other available method: TM-Align.

Conclusions - 1 (Results)

- **Implemented** the 3D Krawtchouk descriptors' computational algorithm, as described by (Mademlis et.al., 2006)
- **Computed** 3D Krawtchouk descriptors for a set of proteins.
- **Created** a simple classifier working based on the features.
- **Evaluated** classification and retrieval performance.
- **Compared** it to retrieval with other available method: TM-Align.

Conclusions - 2 (Details)

- On Kiharalab's computers, running the code for a few weeks, computed the **Krawtchouk moments** for **2434** proteins, for which the voxel grid files were available.
- On Kiharalab's computers, I was running the **TM-Align** algorithm for around a week, and obtained **5924356** comparison scores, ncessary for **ROC analysis**.
- On Condor, I was running the **CE (Combinatorial Extension)** algorithm for a month, and obtained **5924356** comparison scores.
- Using these scores, and features, I evaluated the protein retrieval in terms correct protein class, and summarized it in **precision-recall characteristics**.

Conclusions - 2 (Details)

- On Kiharalab's computers, running the code for a few weeks, computed the **Krawtchouk moments** for **2434** proteins, for which the voxel grid files were available.
- On Kiharalab's computers, I was running the **TM-Align** algorithm for around a week, and obtained **5924356** comparison scores, ncessary for **ROC analysis**.
- On Condor, I was running the **CE (Combinatorial Extension)** algorithm for a month, and obtained **5924356** comparison scores.
- Using these scores, and features, I evaluated the protein retrieval in terms correct protein class, and summarized it in **precision-recall characteristics**.

Conclusions - 2 (Details)

- On Kiharalab's computers, running the code for a few weeks, computed the **Krawtchouk moments** for **2434** proteins, for which the voxel grid files were available.
- On Kiharalab's computers, I was running the **TM-Align** algorithm for around a week, and obtained **5924356** comparison scores, ncessary for **ROC analysis**.
- On Condor, I was running the **CE (Combinatorial Extension)** algorithm for a month, and obtained **5924356** comparison scores.
- Using these scores, and features, I evaluated the protein retrieval in terms correct protein class, and summarized it in **precision-recall characteristics**.

Conclusions - 2 (Details)

- On Kiharalab's computers, running the code for a few weeks, computed the **Krawtchouk moments** for **2434** proteins, for which the voxel grid files were available.
- On Kiharalab's computers, I was running the **TM-Align** algorithm for around a week, and obtained **5924356** comparison scores, ncessary for **ROC analysis**.
- On Condor, I was running the **CE (Combinatorial Extension)** algorithm for a month, and obtained **5924356** comparison scores.
- Using these scores, and features, I evaluated the protein retrieval in terms correct protein class, and summarized it in **precision-recall characteristics**.

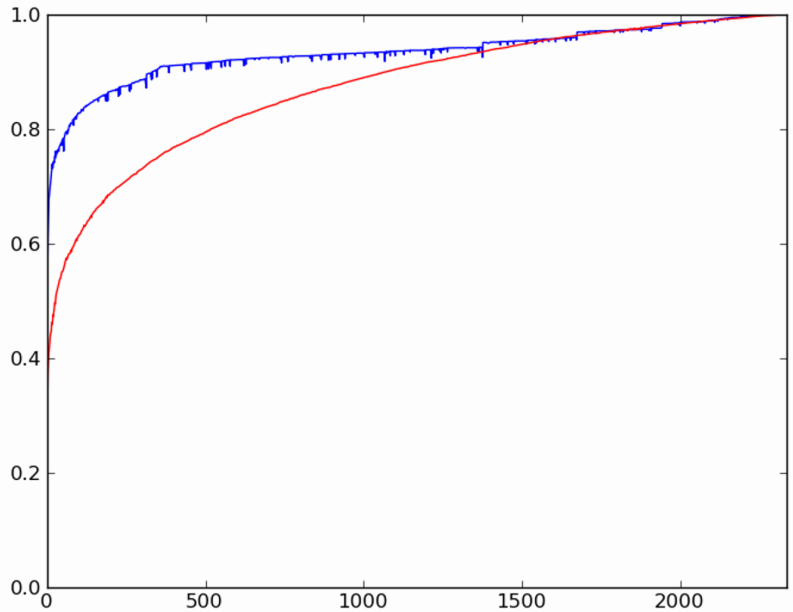
Conclusions - 3 (Details)

Retrieval Summary - Krawtchouk moments vs TM-Align

	Top 1	Top 5	Top 10	AUC
Kawtchouk features	0.80	0.74	0.60	0.878
Full data TM-Align	0.89	0.86	0.73	0.937

Precision-recall - Krawtchouk moments vs TM-Align

Emphasis: the retrieval is worse, but many orders of magnitued faster.



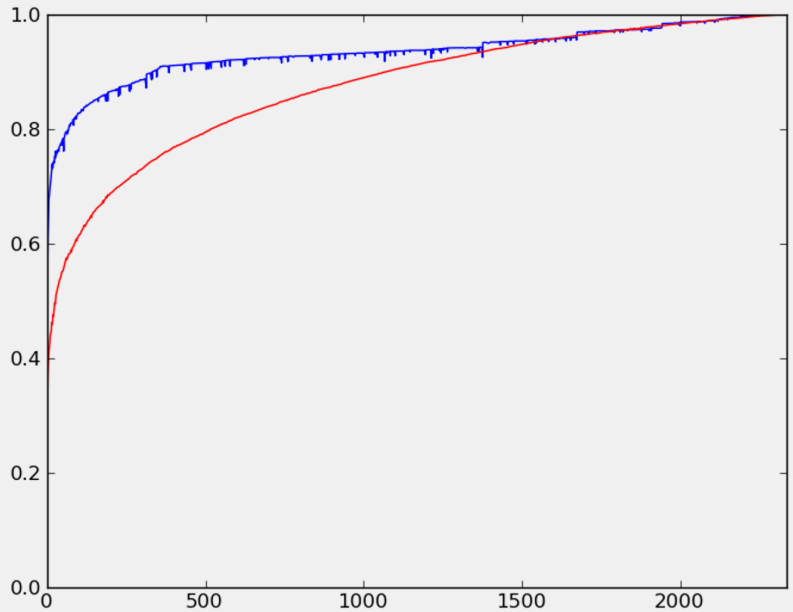
Conclusions - 3 (Details)

Retrieval Summary - Krawtchouk moments vs TM-Align

	Top 1	Top 5	Top 10	AUC
Kawtchouk features	0.80	0.74	0.60	0.878
Full data TM-Align	0.89	0.86	0.73	0.937

Precision-recall - Krawtchouk moments vs TM-Align

Emphasis: the retrieval is worse, but many orders of magnitued faster.



Future work

- **Estimate** ROC confidence intervals.
- **Investigate** local feature extraction (i.e., vary p in $\overline{K}(n, x, p, N)$.)
- **Compare** results directly to Zernike descriptor results.
- **Investigate** rotational invariance of Krawtchouk moments.
- **Extend** the moments to higher dimensions (e.g., protein motion patterns).

Future work

- **Estimate** ROC confidence intervals.
- **Investigate** local feature extraction (i.e., vary p in $\overline{K}(n, x, p, N)$.)
- **Compare** results directly to Zernike descriptor results.
- **Investigate** rotational invariance of Krawtchouk moments.
- **Extend** the moments to higher dimensions (e.g., protein motion patterns).

Future work

- **Estimate** ROC confidence intervals.
- **Investigate** local feature extraction (i.e., vary p in $\overline{K}(n, x, p, N)$.)
- **Compare** results directly to Zernike descriptor results.
- **Investigate** rotational invariance of Krawtchouk moments.
- **Extend** the moments to higher dimensions (e.g., protein motion patterns).

Future work

- **Estimate** ROC confidence intervals.
- **Investigate** local feature extraction (i.e., vary p in $\overline{K}(n, x, p, N)$.)
- **Compare** results directly to Zernike descriptor results.
- **Investigate** rotational invariance of Krawtchouk moments.
- **Extend** the moments to higher dimensions (e.g., protein motion patterns).

Future work

- **Estimate** ROC confidence intervals.
- **Investigate** local feature extraction (i.e., vary p in $\overline{K}(n, x, p, N)$.)
- **Compare** results directly to Zernike descriptor results.
- **Investigate** rotational invariance of Krawtchouk moments.
- **Extend** the moments to higher dimensions (e.g., protein motion patterns).

Acknowledgements

- Prof. Daisuke Kihara
- Dr. Sael Lee
- Postdoc. Hyung Rae Kim
- Postdoc. Bin Li
- Dr. David La
- PULSe

Acknowledgements

- Prof. Daisuke Kihara
- Dr. Sael Lee
- Postdoc. Hyung Rae Kim
- Postdoc. Bin Li
- Dr. David La
- PULSe

Acknowledgements

- Prof. Daisuke Kihara
- Dr. Sael Lee
- Postdoc. Hyung Rae Kim
- Postdoc. Bin Li
- Dr. David La
- PULSe

Acknowledgements

- Prof. Daisuke Kihara
- Dr. Sael Lee
- Postdoc. Hyung Rae Kim
- Postdoc. Bin Li
- Dr. David La
- PULSe

Acknowledgements

- Prof. Daisuke Kihara
- Dr. Sael Lee
- Postdoc. Hyung Rae Kim
- Postdoc. Bin Li
- Dr. David La
- PULSe

Acknowledgements

- Prof. Daisuke Kihara
- Dr. Sael Lee
- Postdoc. Hyung Rae Kim
- Postdoc. Bin Li
- Dr. David La
- PULSe