# Dictionary Learning for Games

**Manny Ko**
Principal Engineer, Activision R&D
Graphics Research and Development

GAME DEVELOPERS CONFERENCE 2014
SAN FRANCISCO, CA
MARCH 17–21, 2014
EXPO DATES: MARCH 19–21

# Outline

- K-SVD and dictionary learning
- Linear Blend Skinning
    - Brief survey on automatic skinning and compression
- Dictionary learning for LBS
    - Two-layer sparse compression of Le & Deng.
- This talk is about compressing skinned animations.

# Frames, Sparsity and Global Illumination: New Math for Games GDC 2012

Robin Green – Microsoft Corp
Manny Ko – PDI/Dreamworks

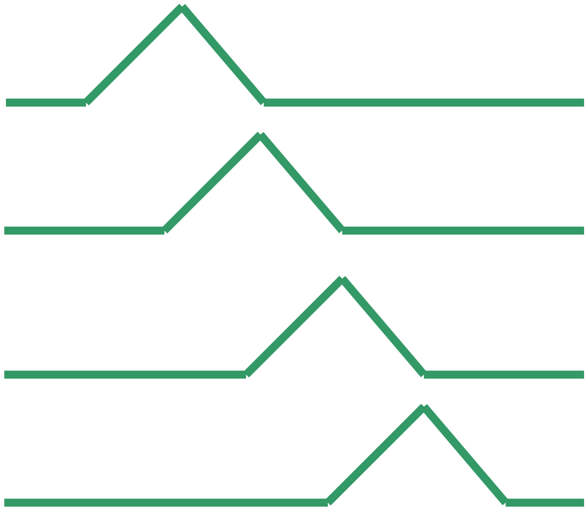
# Orthogonal Matching Pursuit and K-SVD for Sparse Encoding

**Manny Ko**
Senior Software Engineer, Imaginations Technologies
**Robin Green**
SSDE, Microsoft Xbox ATG

GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 25–29, 2013
EXPO DATES: MARCH 27–29
2013

# Representing Signals

- We represent signals as linear combinations of things we already know – the ‘basis’



$$\times\, \alpha_0 + \times\, \alpha_1 + \times\, \alpha_2 + \times\, \alpha_3 + \cdots =$$

# Orthonormal Bases (ONBs)

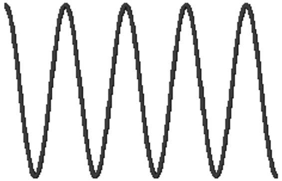
- The simplest way to represent signals is using a set of *orthonormal bases*

$$\int_{-\infty}^{+\infty} b_i(t) b_j(t)\, dt = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

# Example ONBs

- Fourier Basis

$$b_k(t) = e^{i2pkt}$$

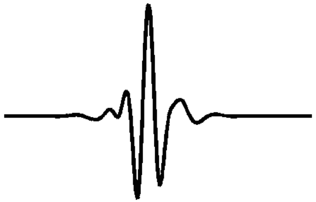


- Gabor Functions

$$b_{k,n}(t) = \omega(t - bn)e^{i2pkt}$$

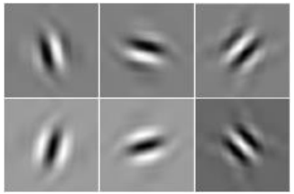
- Wavelets

$$b_{m,n}(t) = a^{-m/2}x(a^{-m}t - bm)$$



- Contourlet

$$b_{j,k,\mathbf{n}}(t) = \lambda_{j,k}\left(t - 2^{j-1}\mathbf{S}_k\mathbf{n}\right)$$



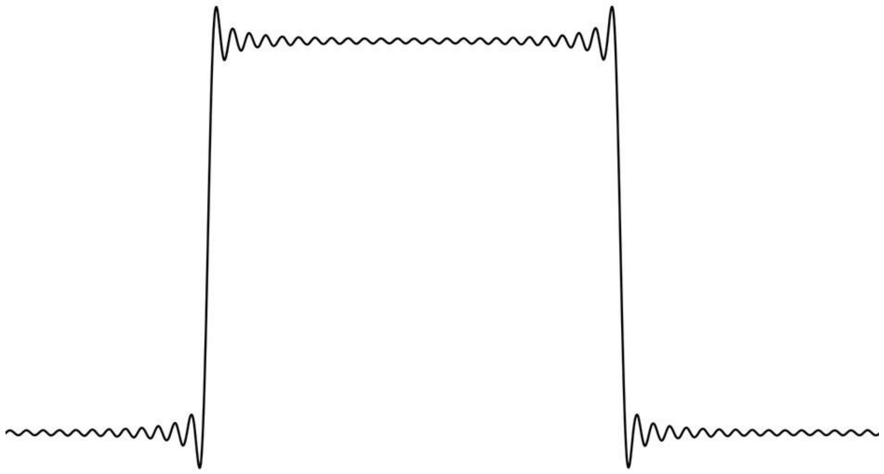
# Benefits of ONB

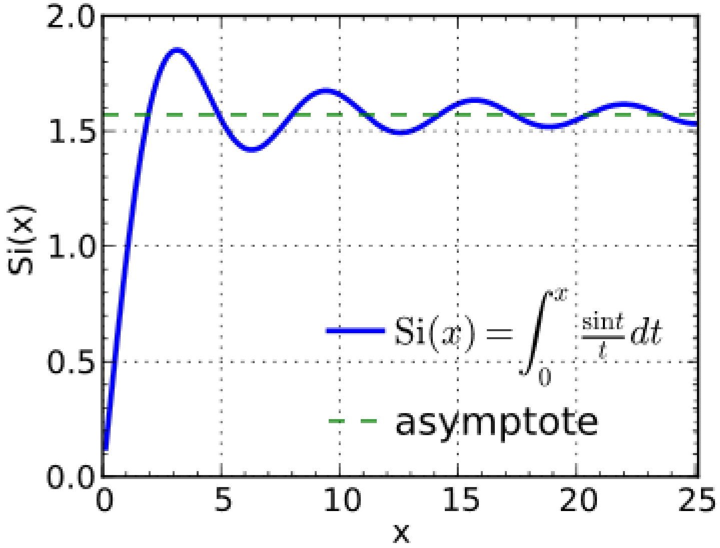
- Analytic formulations
- Well understood mathematical properties
- Fast and simple algorithms for projection

# Problems with ONB

- One-size-fits all – not data adaptive
- Global support cannot adapt to data locally
  - Fourier support is infinite, SH support spans the sphere
  - Try using Fourier to represent a step-function
- Not sparse – very few zero coefficients
- Not additive - relies on destructive cancellation.

# Gibb’s Ringing – Fourier and SH

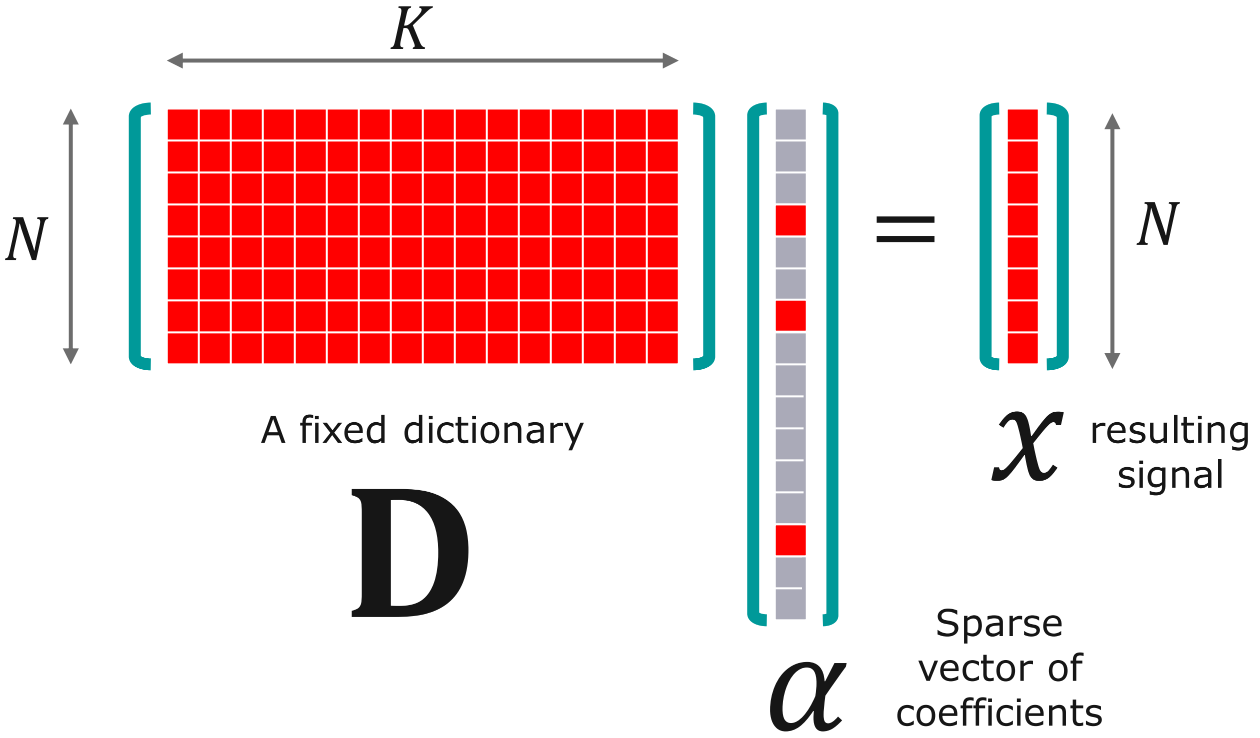




# What is Overcomplete Dictionary?

- Overcomplete means the dictionary has more atoms (columns) than the minimum required for the dimension of the signal
  - In 3D, an ONB only needs 3 basis
  - A 3D dictionary can have dozens or hundreds

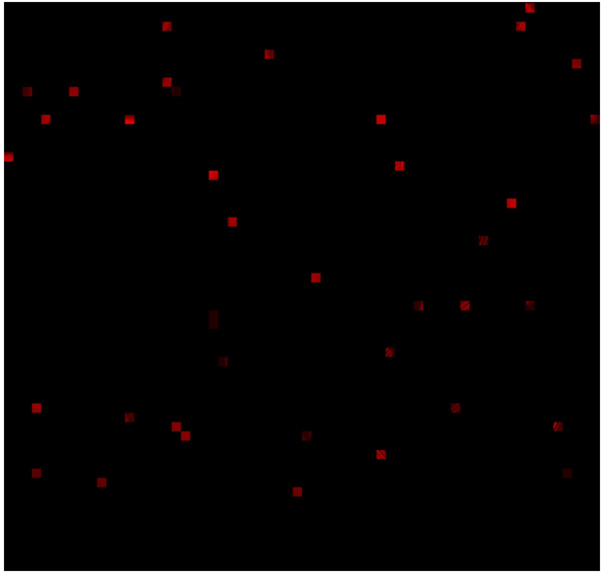
# The Sparse Signal Model



# Why so many atoms?

- More atoms give our algorithm a better chance to find a small subset that matches a given signal
  - Let’s look at some patches from Barbara

# Patches from Barbara



GDC 14

GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 17–21, 2014
EXPO DATES: MARCH 19–21
2014

# Domain Specific Compression

- Just 550 bytes per image

1. Original
2. JPEG
3. JPEG2000
4. PCA
5. KSVD per block

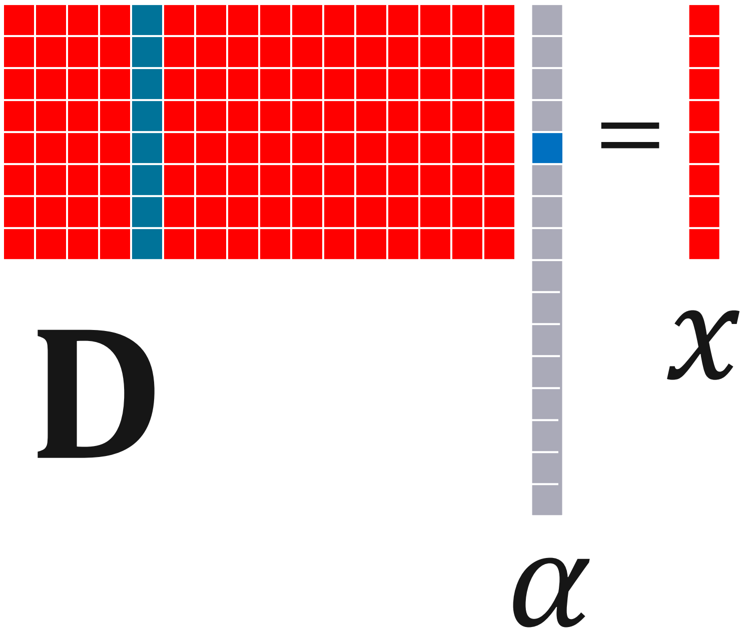
# Project onto Dictionaries

- Overcomplete and non-orthogonal
  - interactions among atoms cannot be ignored
- How do we project?
  - ***Sparse Coding*** *problem*

# Greedy Methods

**Matching Pursuit**

1. Set the residual $r = x$
2. Find an unselected atom that best matches the residual $\|\mathbf{D}\alpha - r\|$
3. Re-calculate the residual from matched atoms $r = x - \mathbf{D}\alpha$
4. Repeat until $\|r\| \leq \epsilon$



# Orthogonal Matching Pursuit (OMP)

- Add an Orthogonal Projection to the residual calculation

1. set $I := \{\emptyset\},\ r := x,\ \gamma := 0$
2. while ($stopping\ test\ false$) do
3. $\quad k := \underset{k}{\operatorname{argmax}} \left| d_k^T r \right|$
4. $\quad I := (I, k)$
5. $\quad \gamma_I := (\mathbf{D}_I)^{+} x$
6. $\quad r := x - \mathbf{D}_I \gamma_I$
7. end while

# What is Dictionary Learning?

- select a few atoms for each signal – e.g. OMP
- Adjust the atoms to better fit those signals
- Repeat

# K-SVD

- Is one of the well known dictionary learning methods
  - Check out our GDC2013 talk
  - our GDC13 slides "OMP and K-SVD for Sparse Coding“
  - See Jim’s talk just before this session
- Miral’s Online Learning is the other.

# Overcomplete Dictionary Recap

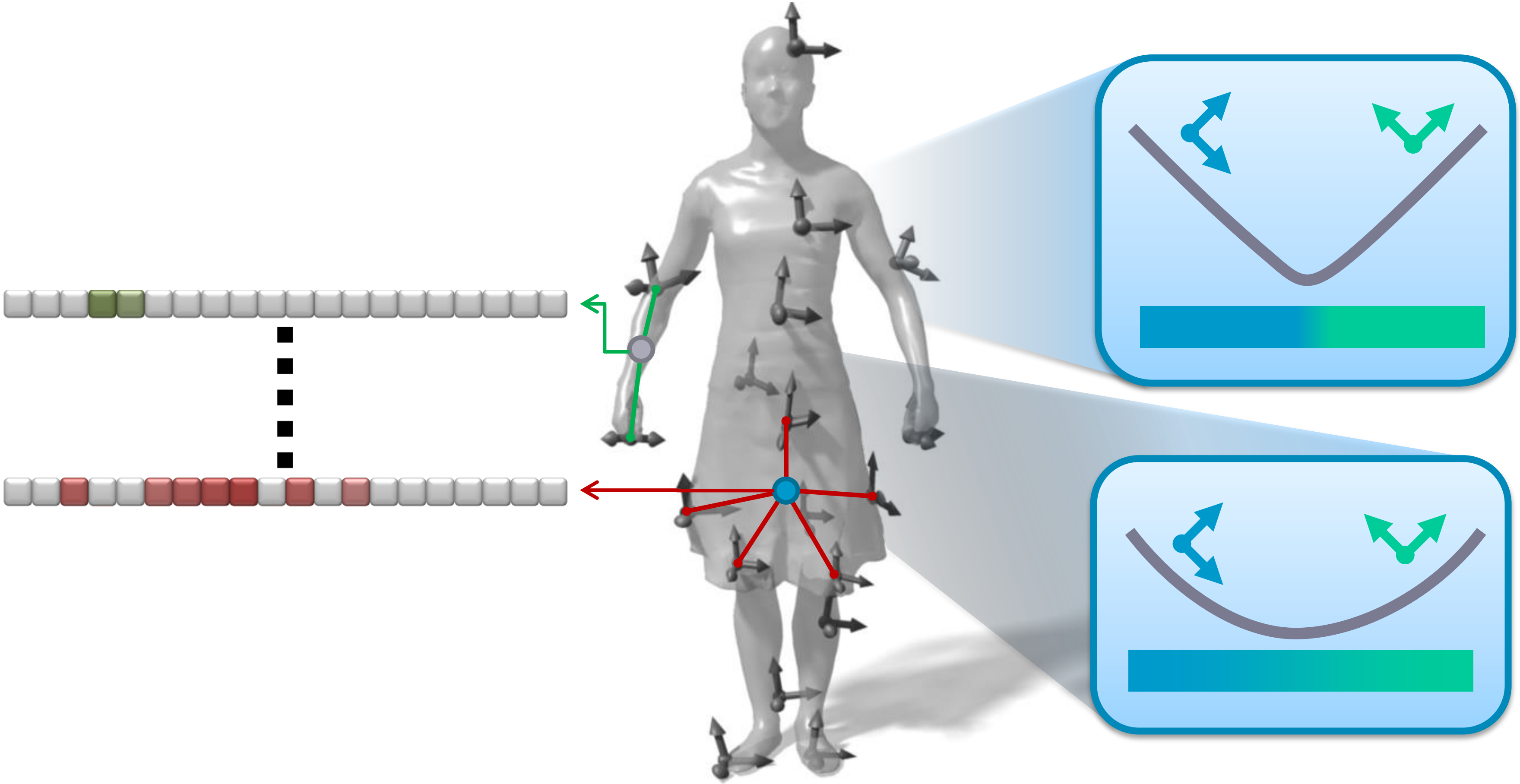
- Importance of overcomplete dictionaries
- OMP for efficient projection onto dictionaries
- K-SVD for learning a better dictionary using samples from the real data

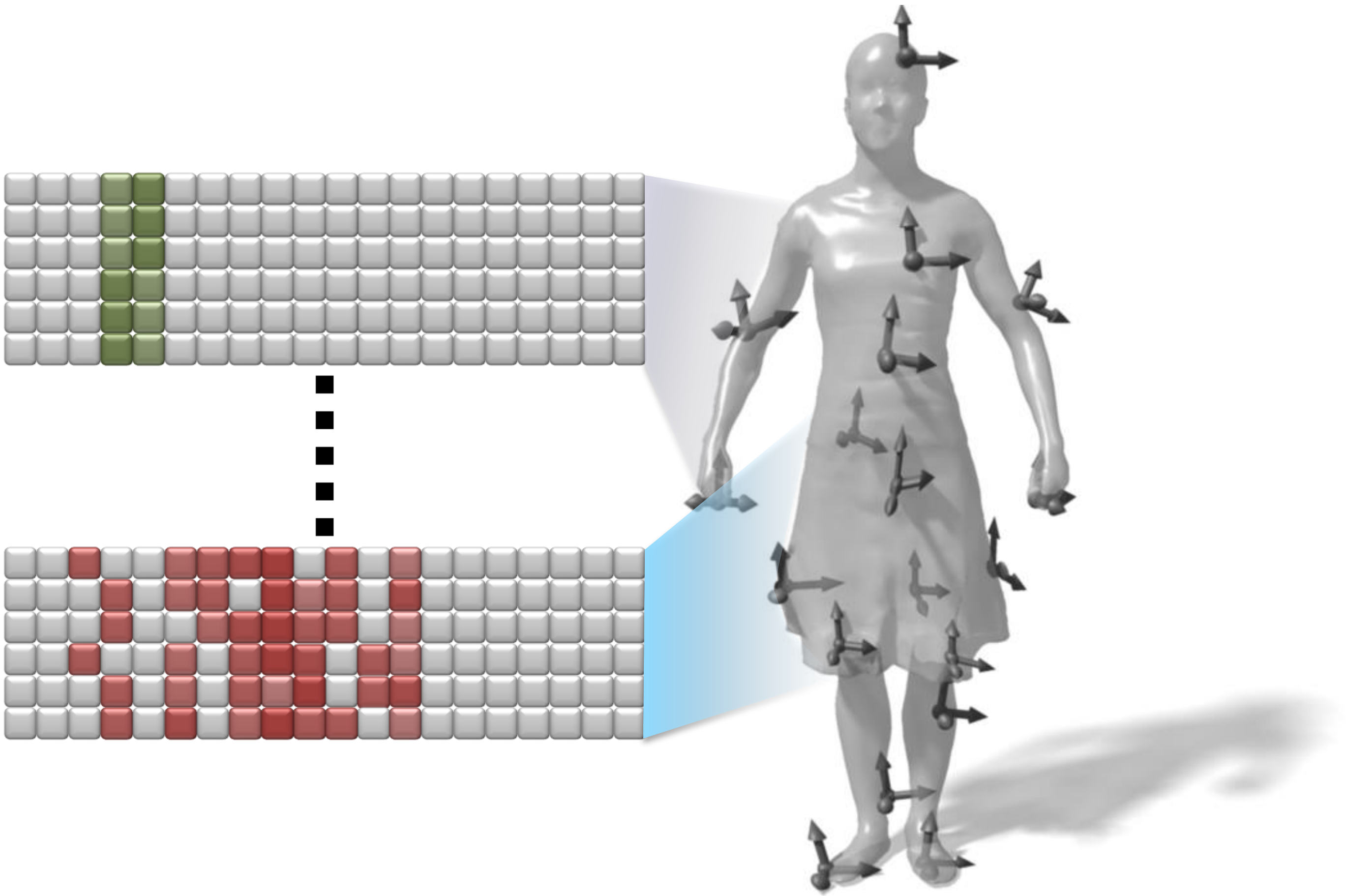
# Part 2: Skinning

- blank

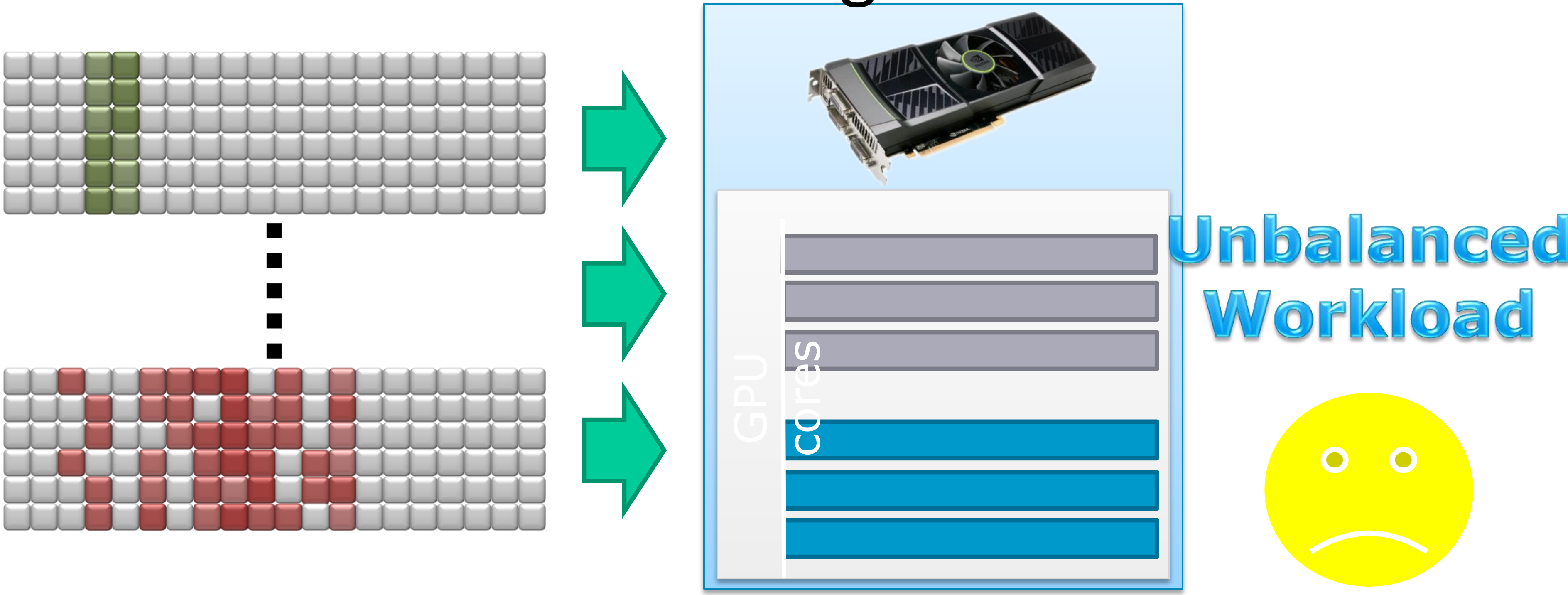
# Linear Blend Skinning

- $v_i = \sum_{j=1}^{|B|} w_{ij}(R_j p_j + T_j)$
  - $p_i$ is the position for the $i$th vertex of the rest pose
  - $w_{ij} \geq 0$ *and sums to one*(affinity). The non-negative constraint makes the blend additive. The affinity constraint prevents over-fitting and artifacts.
  - $R_j$ usually is orthogonal to avoid shearing or scaling
  - $|B|$ is the number of weights (usually <= 6)





# Blend Skinning on GPU



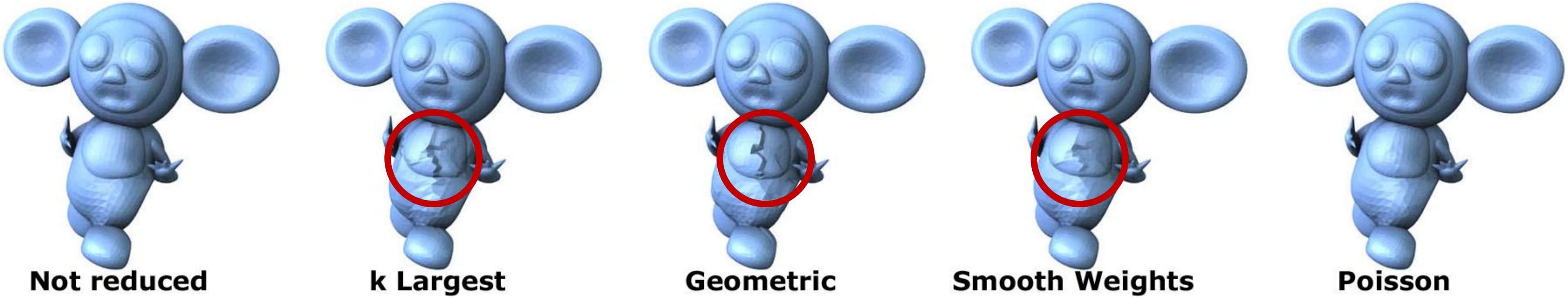
# LBS on GPUs

- $w_{ij}$ typically very sparse – 4-6 weights or less per-vertex
- Ideally a group of vertices all have the same weights to avoid thread divergence or splitting drawcalls
- These are fairly serious constraints

  a) Some vertices might need more weights – e.g. very smooth meshes or complex topology (hand)

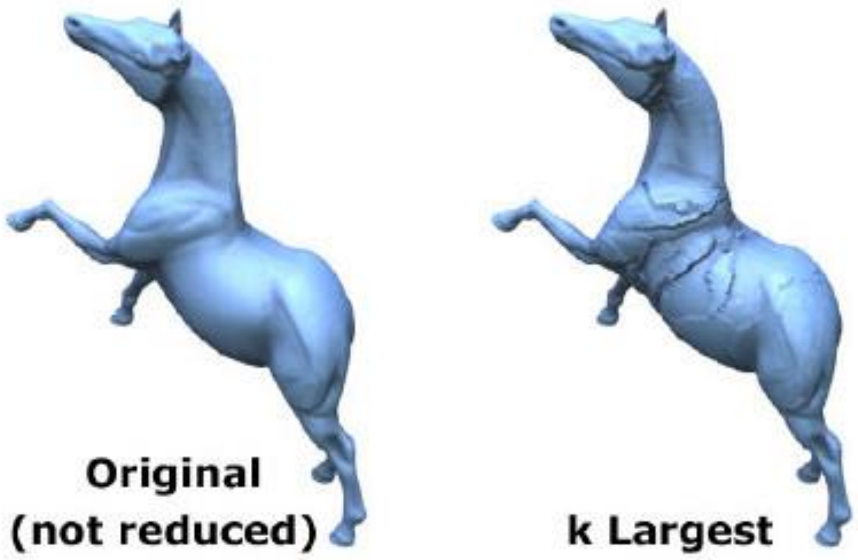
# Weights Reduction

**Poisson-based Weight Reduction of Animated Meshes** [Landreneau and Schaefer 2010]

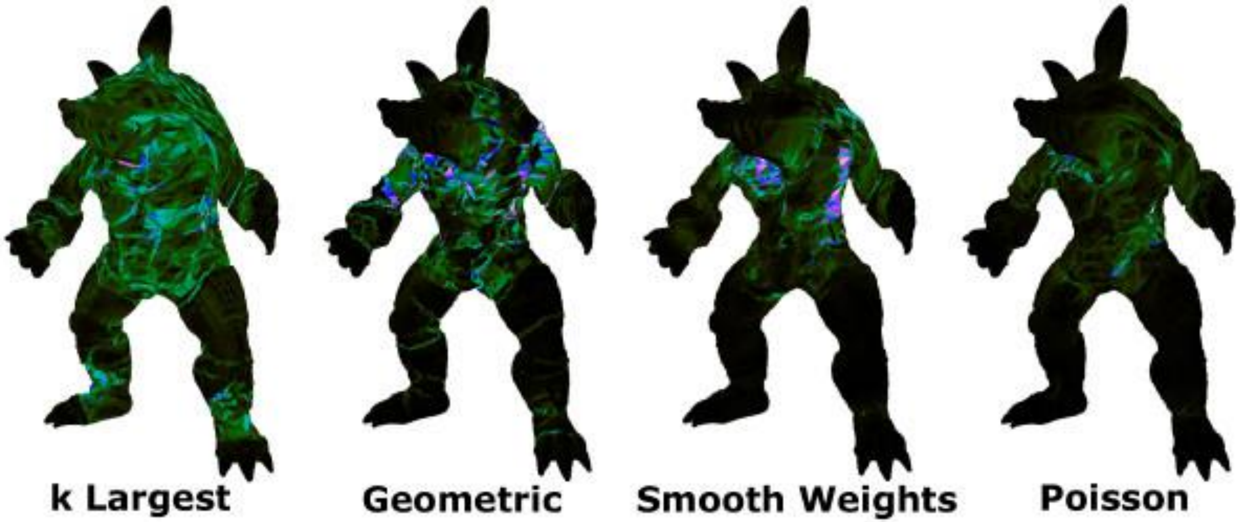



- Discrete optimization: $|\{w_{ij}|w_{ij} \neq 0\}| \leq |K|, \forall i$
  - Impossible to find optimum solution
  - Very **high cost** for non-optimum solution
    - Fracture
    - Significant increase of computing cost: $nK$ non-zero → $n(K+1)$ non-zero

# K-Largest - fracturing



# K-Largest - normals



# Vertex Normal in Shader

$$min \sum_{M} |\alpha_i (M c_i^0 - \hat{c}_i)|^2$$

Solving for the inverse transpose

$$M^{-T} = \frac{1}{\beta} adj(\sum_i \alpha_i^2 c_i^0 \hat{c}_i^T)(\sum_i \alpha_i^2 c_i^0 c_i^T)$$

# Magic 4

- why 4 weights is too few to generate smooth weights
    - 4 vertices specifies an affine transform exactly.
    - simplices in 3D contains 4 vertices for barycentric coordinates.

SIGGRAPH2013

# Two-Layer Sparse Compression of Dense-Weight Blend Skinning

Binh Le and Zhigang Deng
UNIVERSITY of HOUSTON

# Two-Layer Sparse Compression, Le & Deng 2013

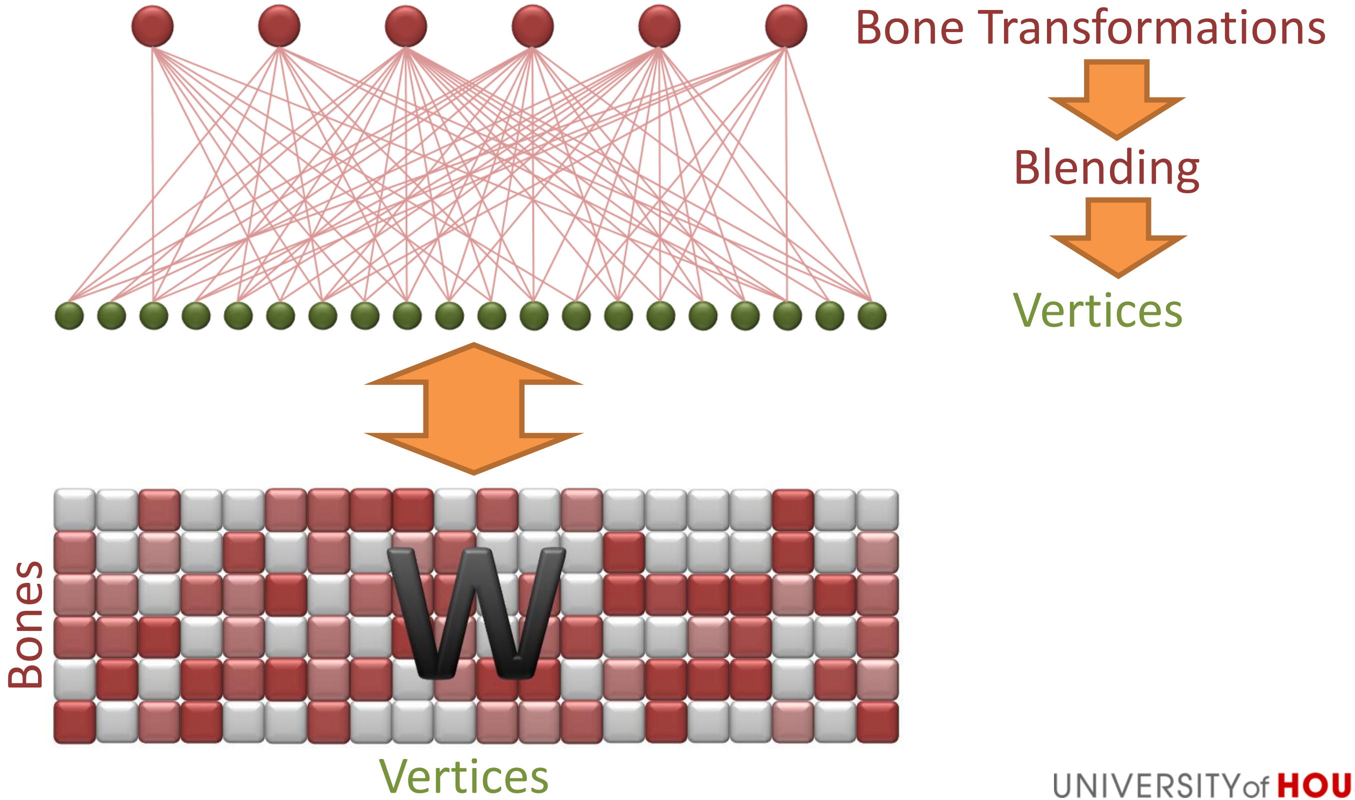
- Use **dictionary learning** to compute a two-level compression using bones
  - Work with the weights of the bind-pose directly

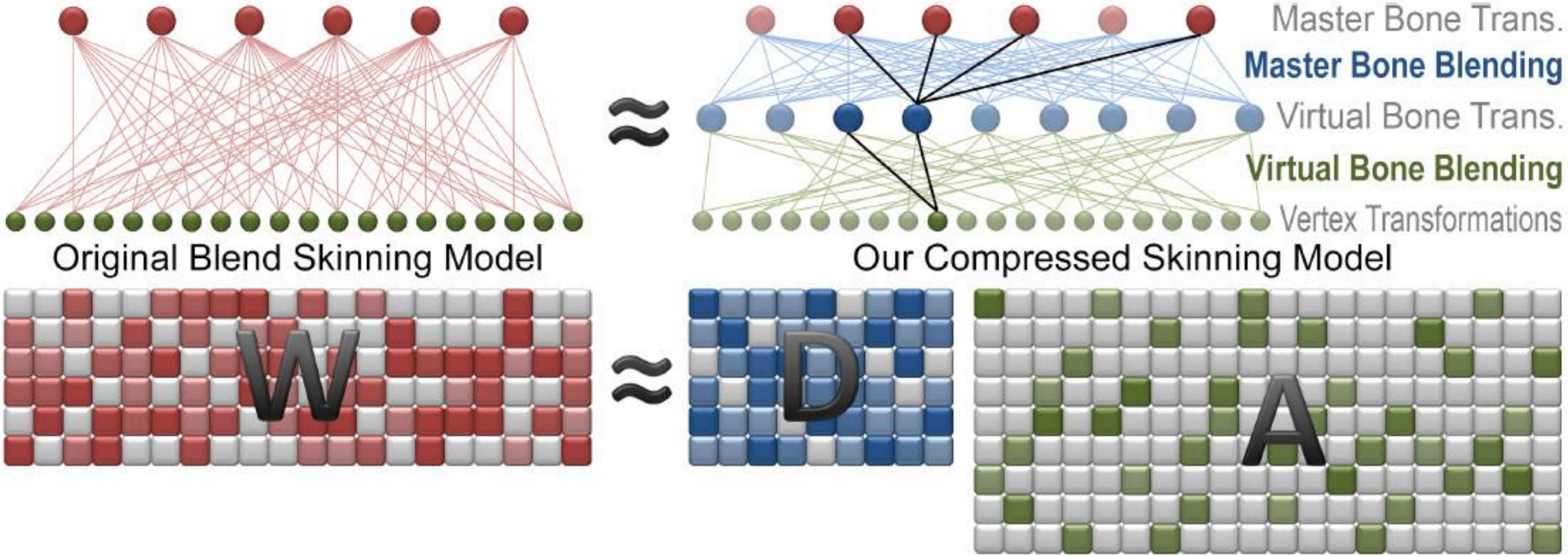
# Why Dictionary for LBS?

- Why dictionary learning?
  - limitations of Orthonormal-basis  e.g. eigen/PCA
    - Not adaptive
    - Not purely additive – i.e. negative weights (relies on cancellation)
    - No intuitive meaning – bones extracted cannot be used to tweak the model

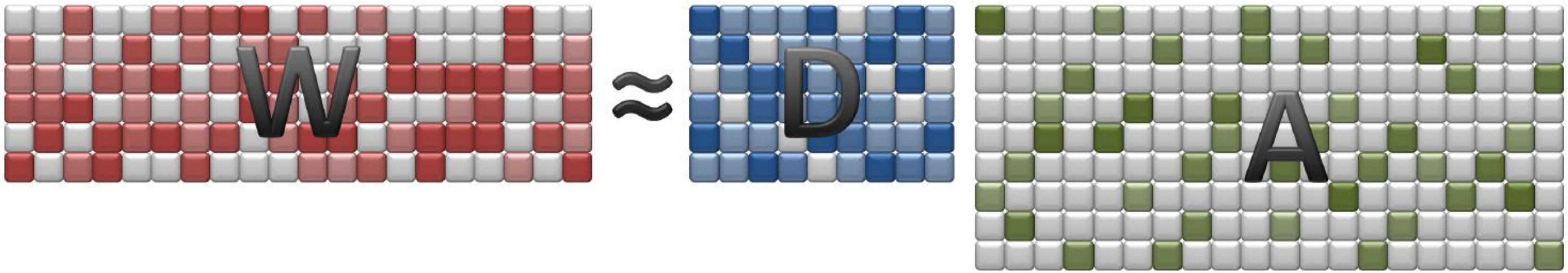
# Dense-Weight Compression

❖ **Input:** Dense matrix




Master Bone Trans.
Master Bone Blending
Virtual Bone Trans.
Virtual Bone Blending
Vertex Transformations
Original Blend Skinning Model
Our Compressed Skinning Model
W
≈
D
A

# Sparse Matrix Factorization – dictionary learning

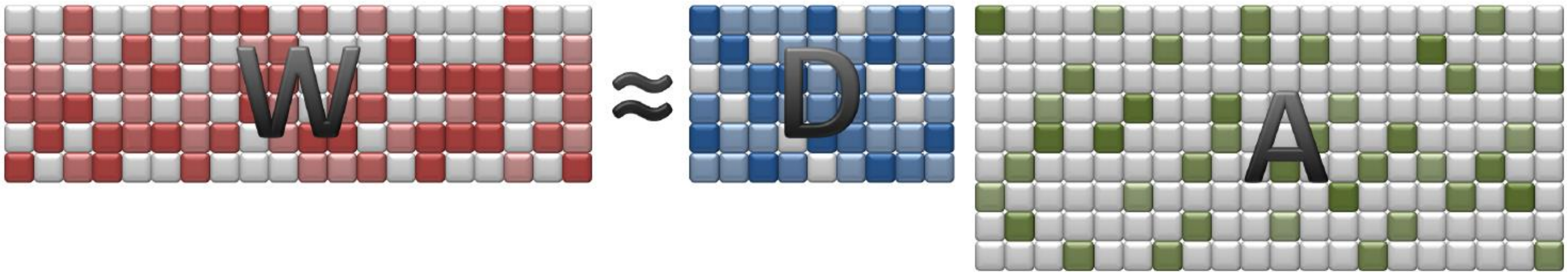



$$\min_{D,A} \Delta_W{}^2 = \min_{D,A} \frac{1}{kn} \|DA - W\|_F^2$$

$$\text{Subject to: } \mathbf{card}(d_i) \leq c, \forall i$$

$$\mathbf{card}(\alpha_i) \leq 2, \forall i$$

# Sparse Matrix Factorization

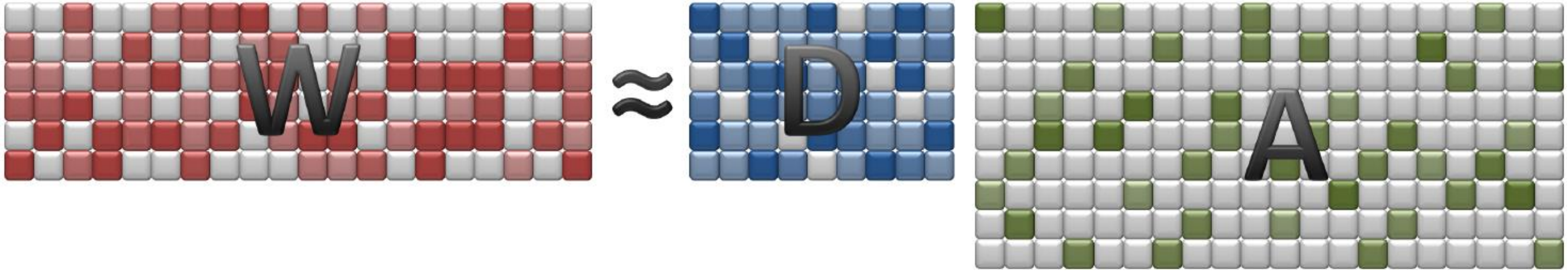



$$\min_{D,A} {\Delta_W}^2 = \min_{D,A} \frac{1}{kn} \|DA - W\|_F^2$$

Subject to: $\mathbf{card}(d_i) \leq c, \forall i$ $\leftrightarrow$ $c = \max\{\mathbf{card}(w_i)\} + 1$

$\mathbf{card}(\alpha_i) \leq 2, \forall i$

# Sparse Matrix Factorization

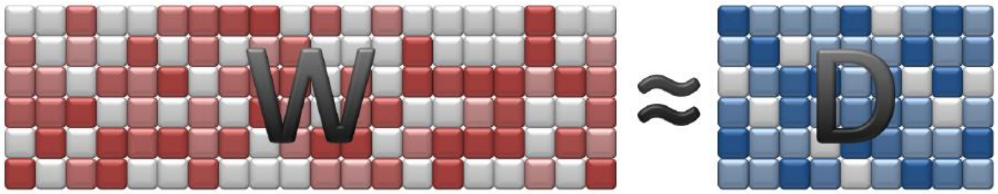


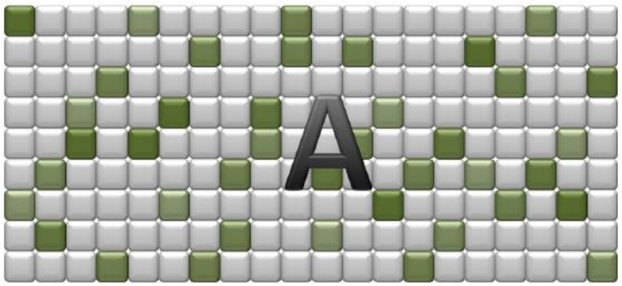

$$\min_{D,A} \Delta_W{}^2 = \min_{D,A} \frac{1}{kn} \|DA - W\|_F^2$$

Subject to: $\mathbf{card}(d_i) \leq c, \forall i \longleftrightarrow c = \max\{\mathbf{card}(w_i)\} + 1$

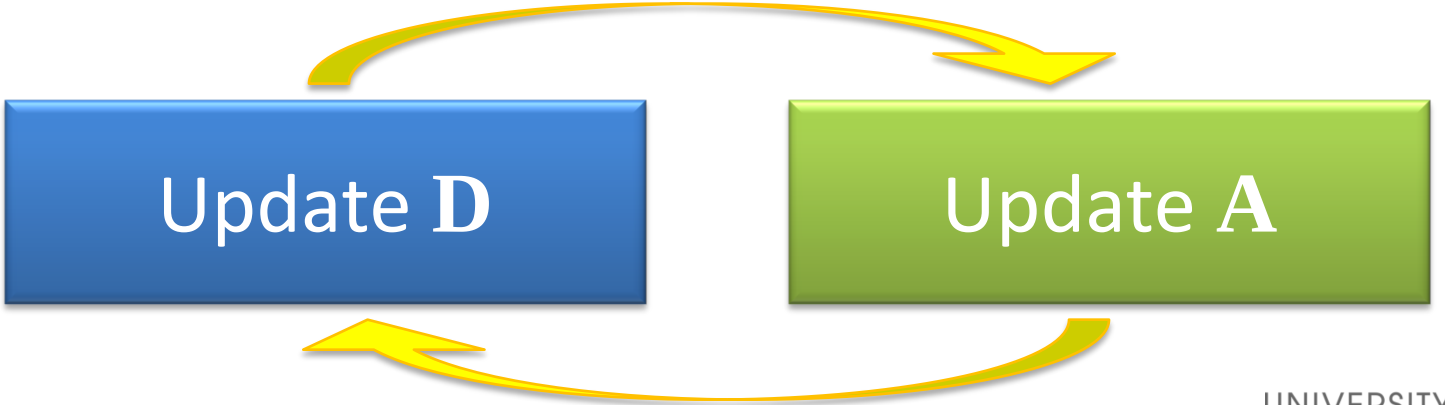
$\mathbf{card}(\alpha_i) \leq 2, \forall i \longleftrightarrow \begin{cases} n \text{ is very large} \\ \mathbf{card}(\mathbf{A}) = 2n \rightarrow \min \end{cases}$

# Algorithm – Block coordinate descent

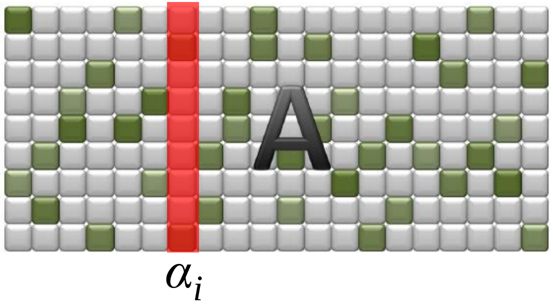



❖ Alternative update $\mathbf{D}$ and $\mathbf{A}$ (Block coordinate descent)



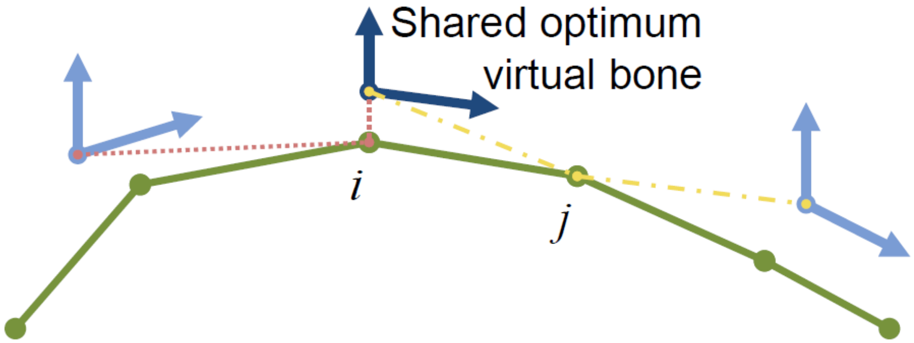
# Update Coefficients A




- Linear least square with 2 unknowns

$$\min_{\substack{(\alpha_i)_r \\ (\alpha_i)_s}} \|d_r(\alpha_i)_r + d_s(\alpha_i)_s - w_i\|_2^2 \text{ s.t. } (\alpha_i)_r + (\alpha_i)_s = 1$$

- Use mesh smoothness assumption to quickly find the non-zero candidates (virtual bones)



# Reduction without Example Pose

# Analysis of Two-Layer Scheme

- Use 100's of virtual bones means we are not limited to a sparse approximation to the original animation.
- virtual bones act as a 'common subexpression'
  - e.g. think compute shader that writes to LDS.
- Still enforce sparsity on VBs to control runtime cost and LDS usage – but *k* can be 100's.
- Per-vertex weights are
  - very sparse (2 per vertex) and the same for all vertices
  - good for GPU.

# Learning Virtual Bones

- Virtual bones are learned from the dense vertex weights by block-coordinate-descent (BCD):

  Sparse coding: search for a few good atoms among the input columns. Use that to project all the rest of the inputs.

  - Atom update: given the sparse weights from above we seek to adjust the atoms to make them fit the inputs that needs them better – a series of small LS problems.

- Similar to EM/Lloyd-Max

# Sparse Coding

Sparse coding:

- insert the vertex with the largest L2 norm
- add a few more vertex which has the smallest dot-product with the 1st atom
- solve the basis-pursuit with OMP (see K-SVD) or LARS.
- solve 2x2 least-square prob. for $w_{ij}$ to blend masters bones

# Weight Map – matrix A

- Weights and indices for each vertex to blend virtual bones
- solving a small 2x2 linear system to minimize MSE:
  - $\arg min_x \|Dx - w_i\|$^2
- runtime per-vertex cost is just 2 dotp
- no bone hierarchy to worry about
- no warp divergence even for high valence vertices

# Atom Updates

Atom update:

```
foreach vertex
```

- update each atom to minimize error for the set of vertices that reference it (this is like K-SVD)

- Miral’s Online Dictionary Learning [Miral09]

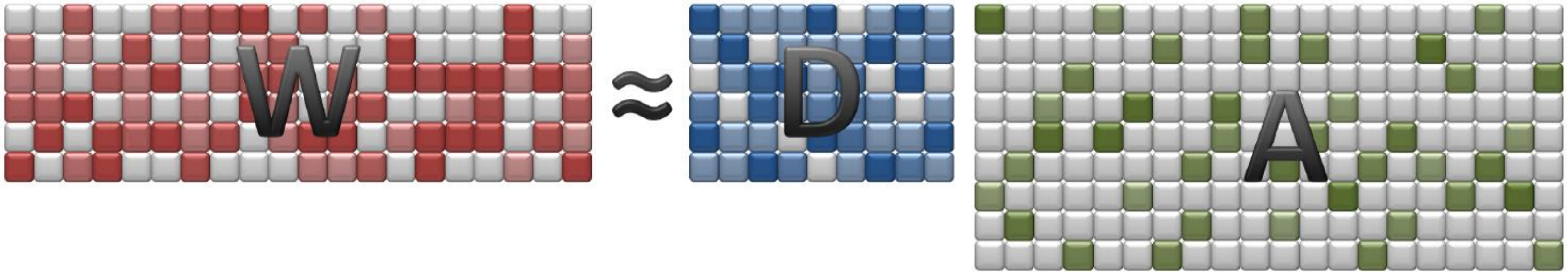
# Atom Updates

- Precompute A and B
  - $A = \sum_{i=1}^{t} \alpha_i \alpha^T$
  - B = $\sum_{i=1}^{t} x_i \alpha^T$
- For all atoms
  - $u_j \leftarrow \frac{1}{A_{j,j}}\left(b_j - D a_j\right) + d_j$ $-$ eq(5)
  - $d_j \leftarrow \frac{1}{\max(\|u_j\|_2, 1)} u_j.$ $-$ eq(6)
- $u_j$ is thresholded to make sure # of non $-$ zero is below the # of master bones

# Live Demo

- youtube

# Compression with Example Poses




## ❖ Without using example pose

– Minimize weights difference

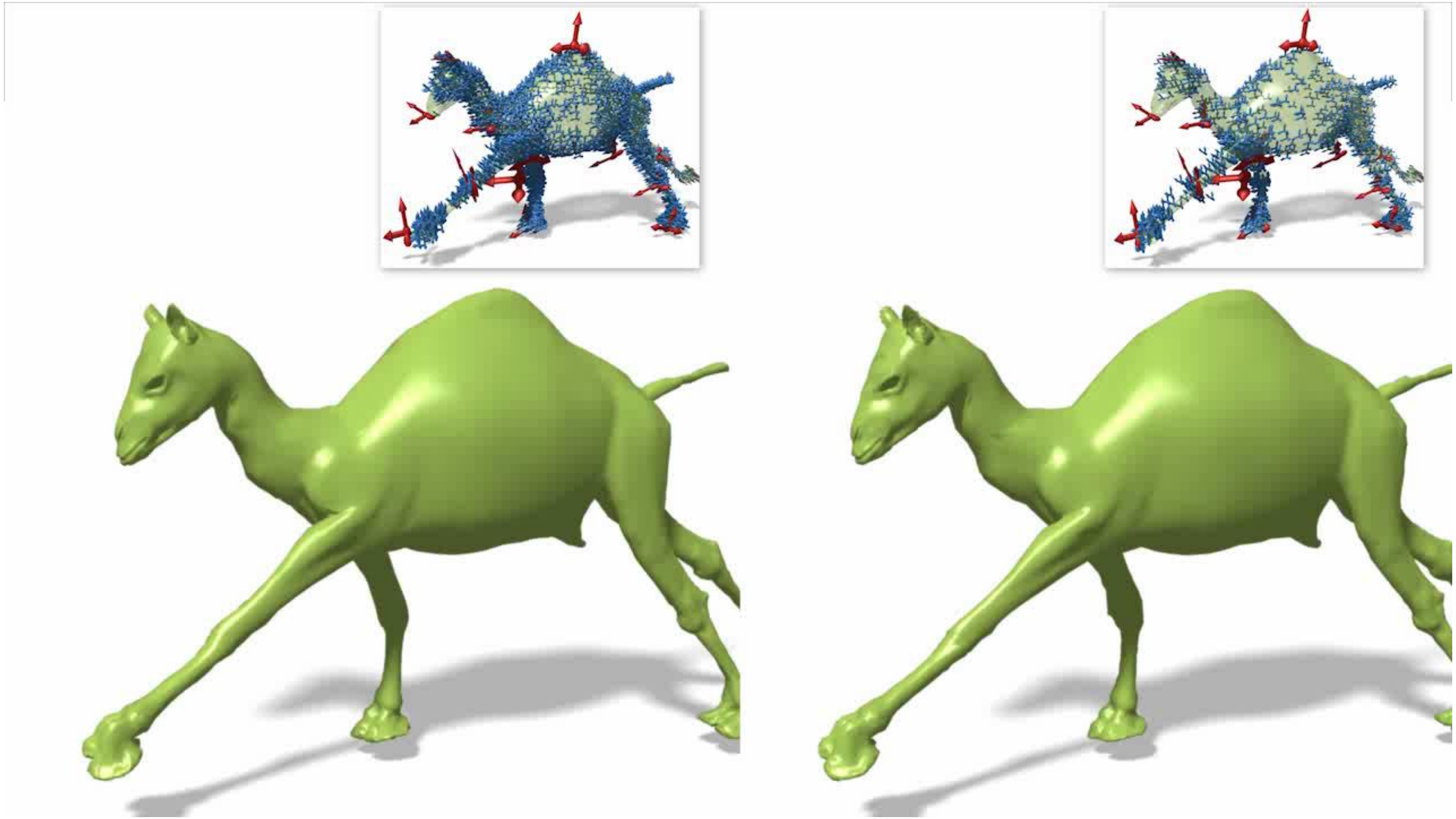
$$\min_{D,A} \frac{1}{kn} \|DA - W\|_F^2$$

## ❖ With using example poses

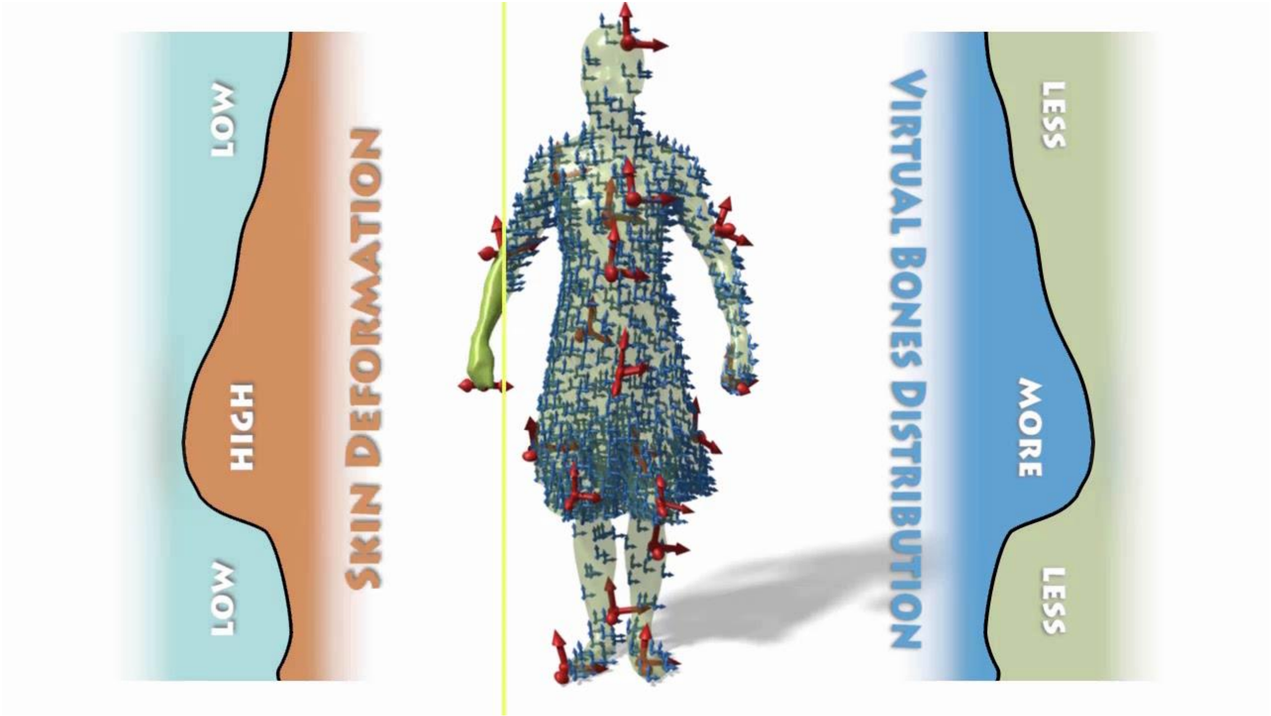
– Minimize reconstruction error

$$\min_{D,A} \frac{1}{3fn} \sum_{i=1}^{n} {E_i}^2$$

# Using Exemplar poses



# Virtual Bones Distribution



# Recap

- The two-level scheme can work with dense (hand painted) weights or example poses (blend shape?)
  - Only the vertex positions are needed
- a fixed memory footprint and uniform per-vertex cost - GPU friendly
- Combines the quality of dense skinning and the efficiencies of sparse-LBS. Animators can use blend-shapes or FFD more.

# Recap 2

- Besides it uses **dictionary learning** and modern **sparsity methods** – how cool is that? ☺
- Last year we show how good dictionary learning is for compressing 2d images and 3d volumes
- Now we see what it can do for animation.
- Thank you!

# Recap 3

- Non-negative LS and Active-set Method (ASM)
- Block-coordinate descent
- Sparsity constraints
  - L1 relaxation and L0-norm constraints
  - Direct solving
- These are all very useful tools.

# Acknowledgements


- Binh Huy Le & Zhigang Deng kindly provided the demo and their Siggraph materials.
- Robin Green for being my collaborator for many years.
- Igor Carron inspired me to learn sparsity methods and matrix factorization and for his spirit of broad exploration and sharing.
- Julien Mairal for the online learning math
- Peter-Pike who inspired me to apply modern math to graphics and games.
- Carlos Gonzalez Ochoa for sharing his insight in animation.

# Activision R&D is Hiring

- Our group is hiring ☺

# References


- Alexa 2000. “As-rigid-as-possible shape interpolation”, SIGGRAPH 2000.
- Halser 2010. “Learning skeletons for shape and pose”, I3D 2010.
- Kavan, Sloan and O'Sullivan 2010. “Fast and Efficient Skinning of Animated Meshes” Comput. Graph. Forum.
- Ko, and Green 2013 “Orthogonal Matching Pursuit and K-SVD for Sparse Encoding” GDC, Math for Games 2013 gdc2013-ompandksvd
- Landreneau & Schaefer “Poisson-Based Weight Reduction of Animated Meshes”, CGF 28(2), 2012.
- Le & Deng 2012. “Smooth skinning decomposition with rigid bones”, ACM TOG, Vol. 31, No. 6.
- Le & Deng 2013. “Two-Layer Sparse Compression of Dense-Weight Blend Skinning”, Siggraph 2013 Paper page
- Mairal 2009. “Online dictionary learning for sparse coding” Int. Conf. on Machine Learning.

# Appendix

- Kabsch/Procrutes method – use SVD to compute the MSE minimum rotation of one point-set to another.
- Kabsch_algorithm