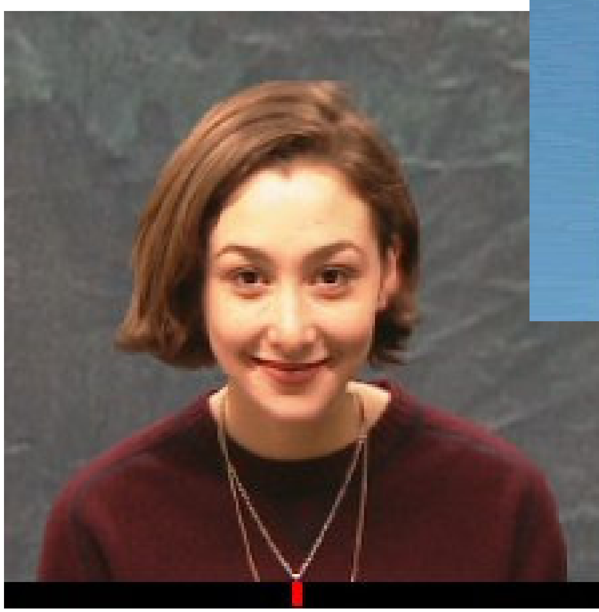
Video Textures

Arno Schödl
Richard Szeliski
David H. Salesin
Irfan Essa



presented by Marco Meyer

Motivation

- Images
  - lack of dynamics
- Videos
  - finite duration
  - lack of „timeless“ quality of image

Motivation

Image	↔	Video Texture	↔	Video
endless	↔	endless & in motion	↔	in motion

Applications:

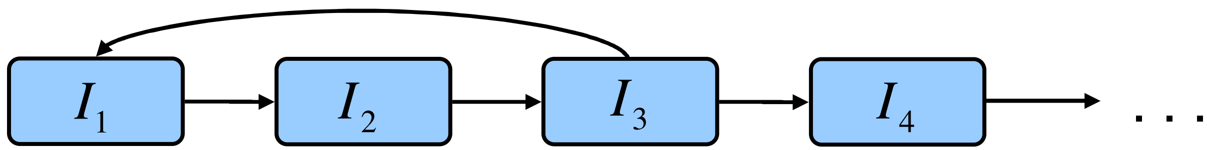
- dyn. scenes on the web
- dyn. background
- user-controlled animation

Outline

- Introduction
- Representation
- MAIN SYSTEM
- Extensions
- Results
- Future Work & Discussion

Introduction

- If two frames are similar, we can create a transition

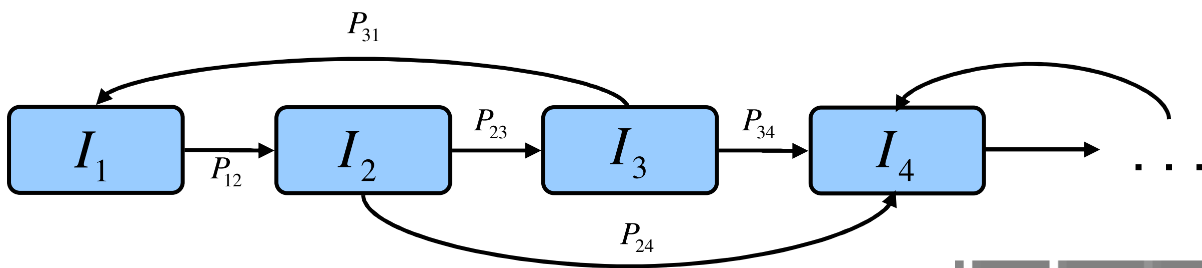


- Find sequence with respect to global structure

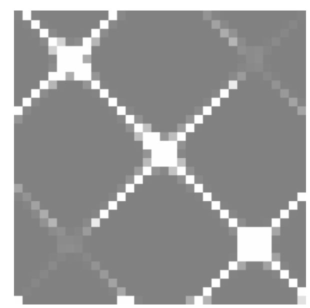
- Smooth the selected transitions

Representation

- VT can be seen as Markov processes



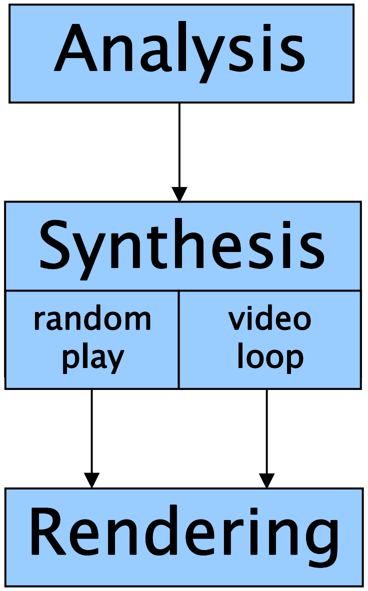
- Matrix of probabilities
  - matrix mostly sparse



- Set of explicit links with associated probability

Outline

- Introduction
- Representation
- MAIN SYSTEM
- Extensions
- Results
- Future Work & Discussion

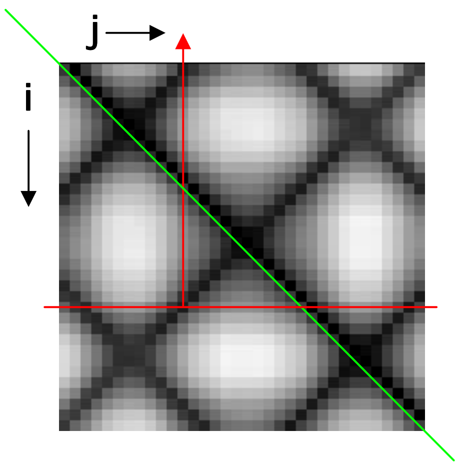


Analysis

- Preparation steps
  - brightness equalization
  - video stabilisation

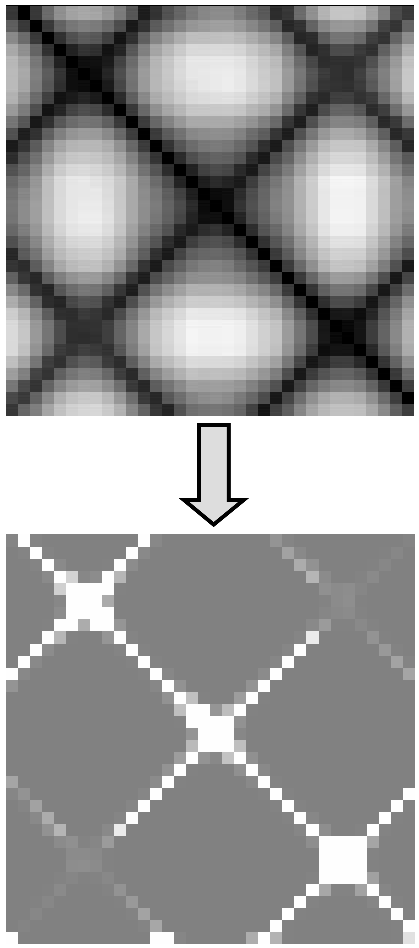
- We need a measure of similarity
  - L_2 distance
  - → distance-matrix D

$$D_{ij} = \left\| I_i - I_j \right\|_2$$



Probability matrix

D_{ij}



P_{ij}

- Map distances to probabilities

$$P_{ij} \propto \exp\left(-\frac{D_{i+1,j}}{\sigma}\right)$$

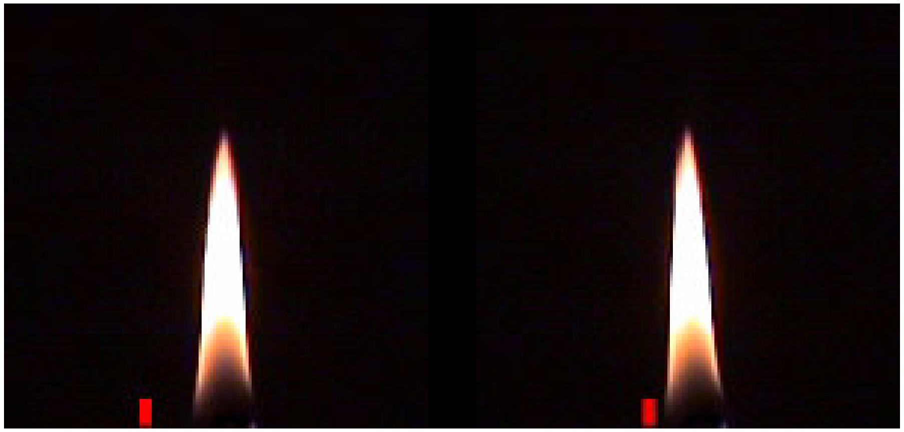
- Normalize so that

$$\sum_j P_{ij} = 1$$

Analysis

Video

$$P_{ij} \propto \exp(-\frac{D_{i+1,j}}{\sigma})$$



high σ   low σ

Preserving Dynamics

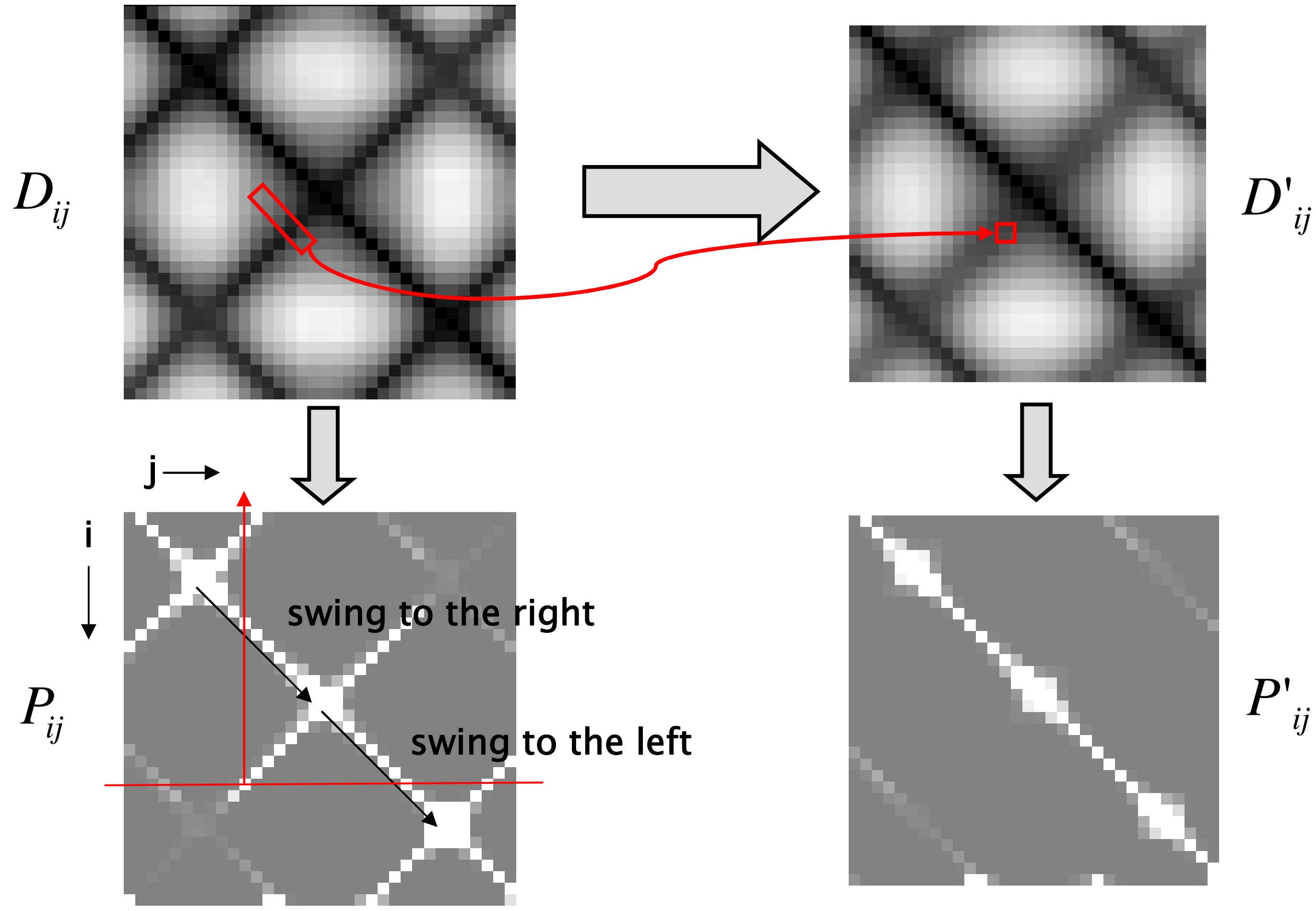
Preserving Dynamics

- Optical flow computation
  - bad in untextured areas

- Compare temporally adjacent frames
  - match subsequences instead of single frames
  - filter matrix with a diagonal kernel with weights w_k

$$D'_{ij} = \sum_{k=-m}^{m-1} w_k D_{i+k,\,j+k}$$

Preserving Dynamics



Dead Ends

- Only window is considered
  - →we might get into dead end
  - →try to predict future cost
- Anticipated future cost
  - propagate future costs backward

$$D''_{ij} = (D'_{ij})^p + \alpha \cdot \sum_k P''_{jk} D''_{jk}$$

with

$$P''_{ij} \propto \exp\left(-\frac{D''_{i+1,j}}{\sigma}\right)$$

- Solve by iteration

Dead Ends

- Only window is considered
  →we might get into dead end
  →try to predict future cost
- Anticipated future cost
  - propagate future costs backward

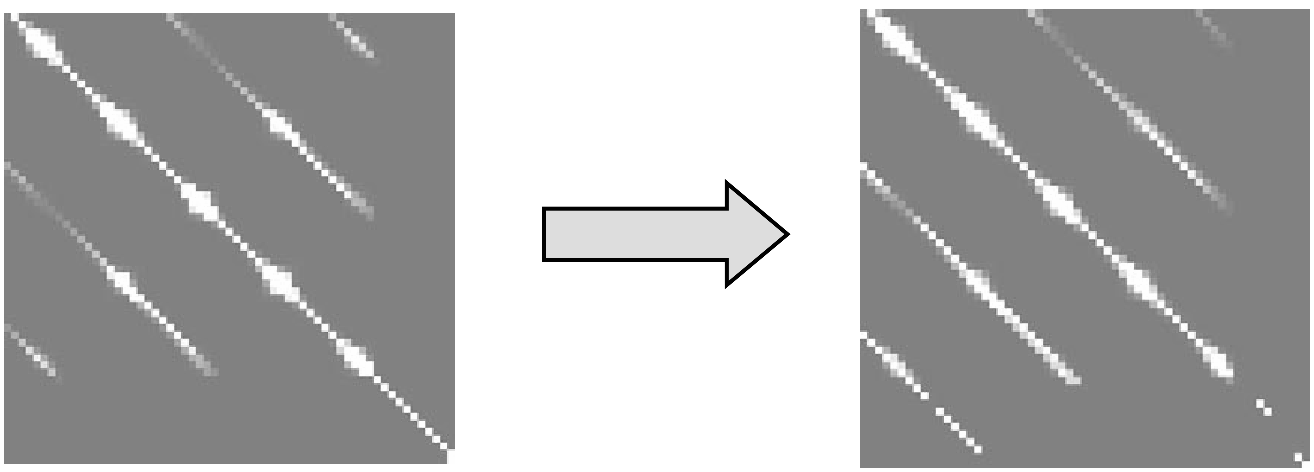
$$D''_{ij} = (D'_{ij})^p + \alpha \cdot \min_k D''_{jk}$$

- Solve with Q-Learning

Dead Ends

- Most effective if solved from last to first row

P'_{ij}



P''_{ij}

Pruning

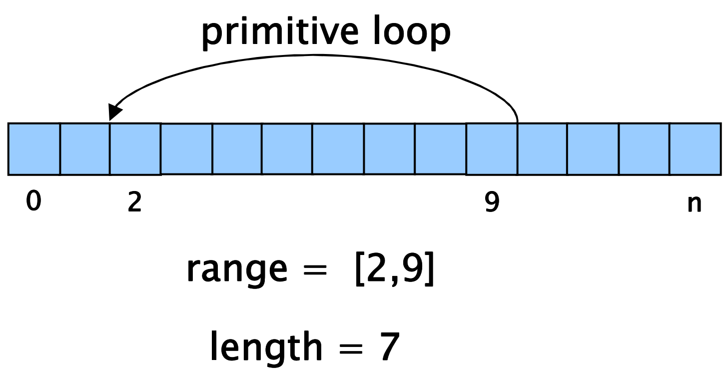
- Goal
  - to supress non-optimal transitions
  - to save storage space
- Paradigms
  1. keep only local maxima
  2. keep those transitions with probabilities above threshold

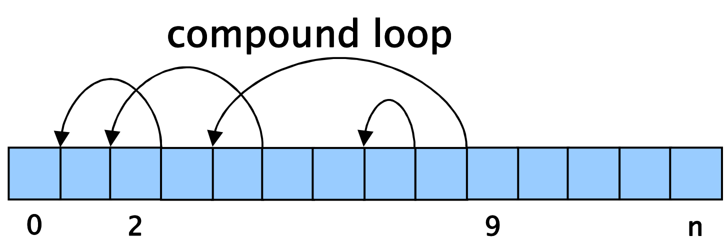
Synthesis

- Random play
  - begin anywhere but in dead end
  - select next frame according to P''_{ij}
  - VT is never repeated exactly the same way

- Video loop
  - fixed length loop
  - for conventional digital video players

Video loop





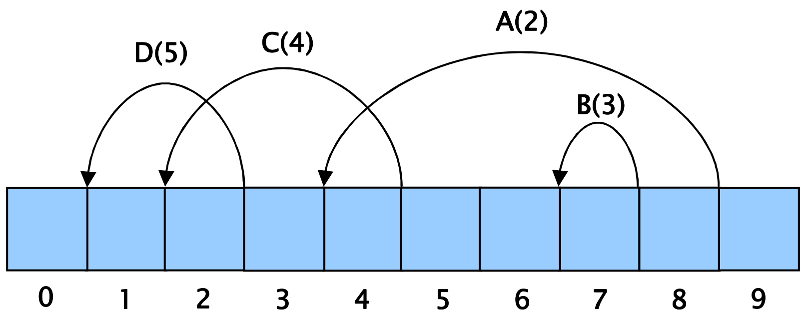
- To combine loops
  → ranges must overlap
- forward transitions not considered
- Loop length

$$\sum_i \#exec_i \cdot length\_of\_loop_i$$

Optimal loop algorithm

- Multiset describes loop with focus to:
  - length
  - range
- Straightforward approach:
  - exponentially in #transitions
- Optimal loop algorithm:
  - select set of transitions (dyn. programming)
  - schedule primitive loops

Select set of transitions

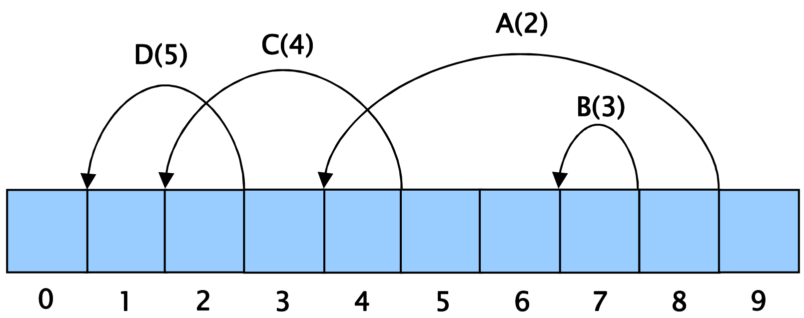


L	A(2)	B(3)	C(4)	D(5)
1				
2				
3				
4				
5				
6				

Alg:

- foreach cell
    - if L>=length of transition of current column
        - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

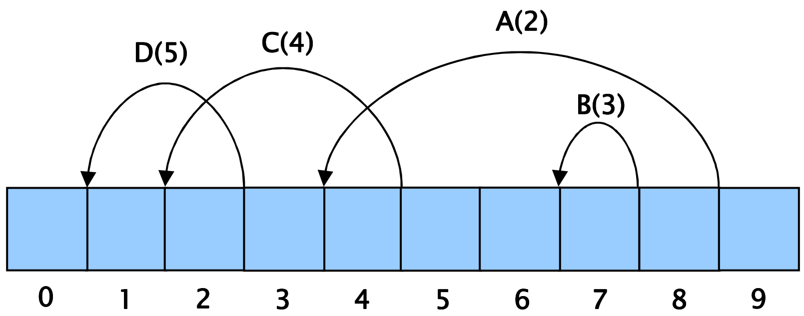


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3				
4				
5				
6				

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

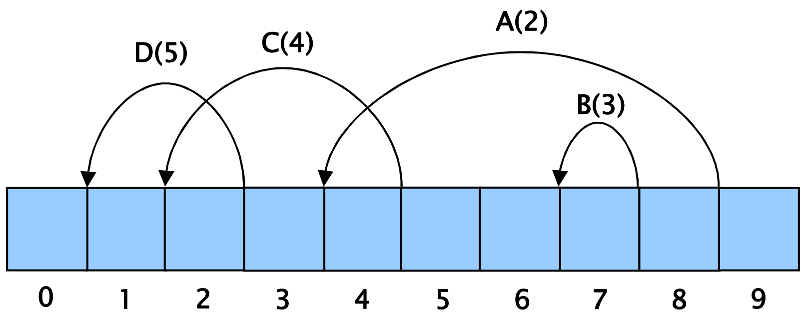


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3				
4				
5				
6				

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

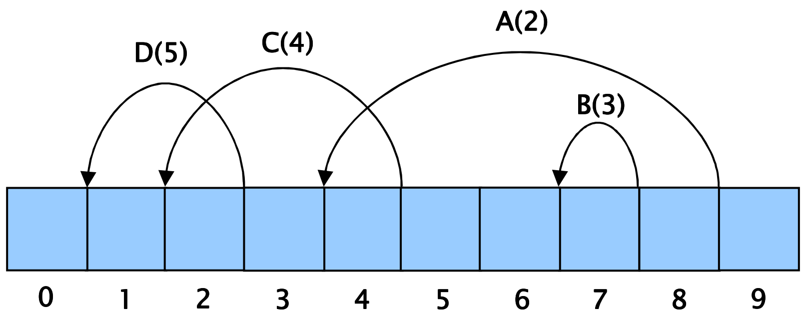


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)		
4				
5				
6				

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

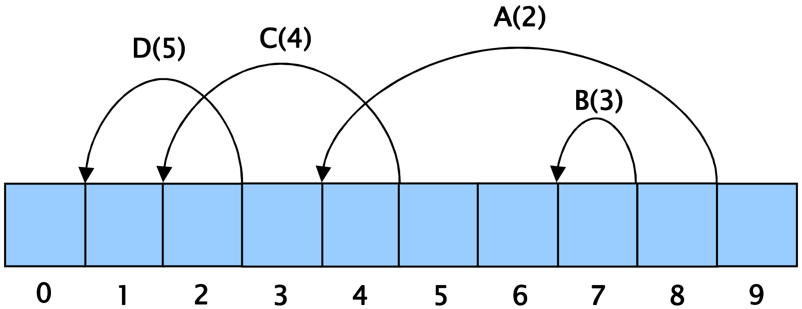


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)			

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

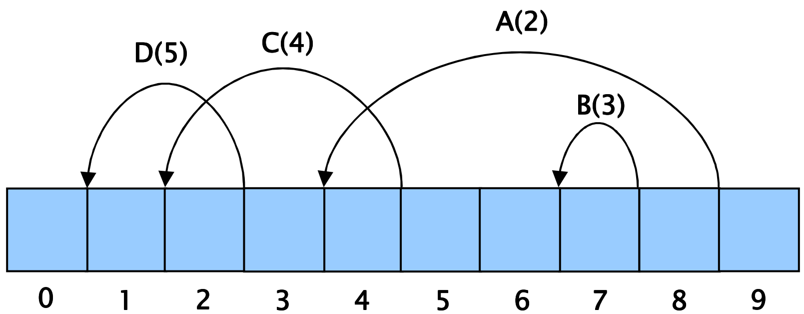


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)			

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

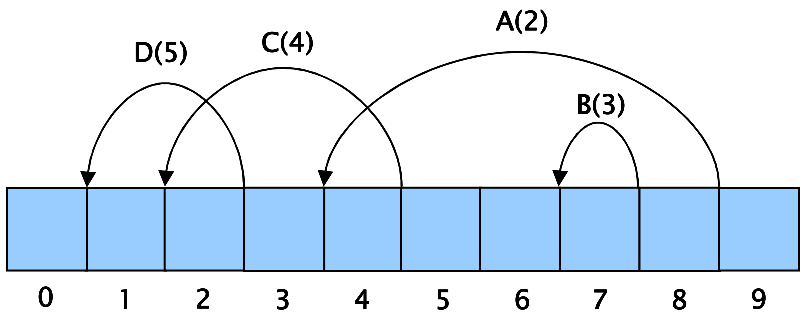


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)			

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

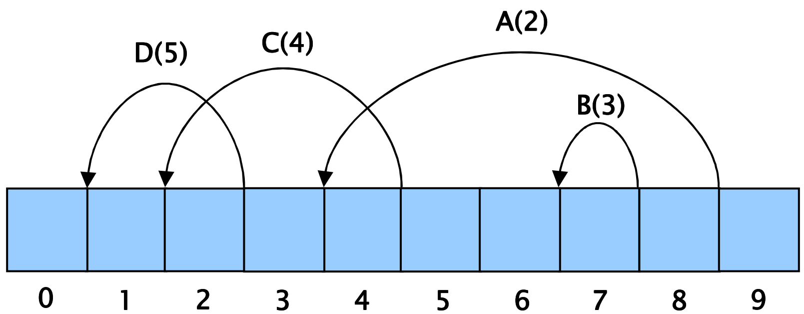


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)			

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

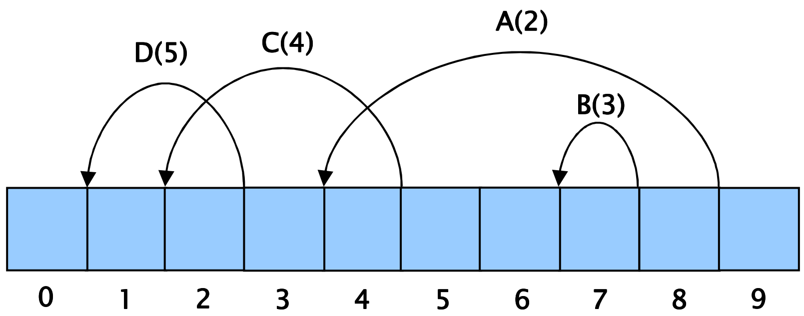


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)			

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions

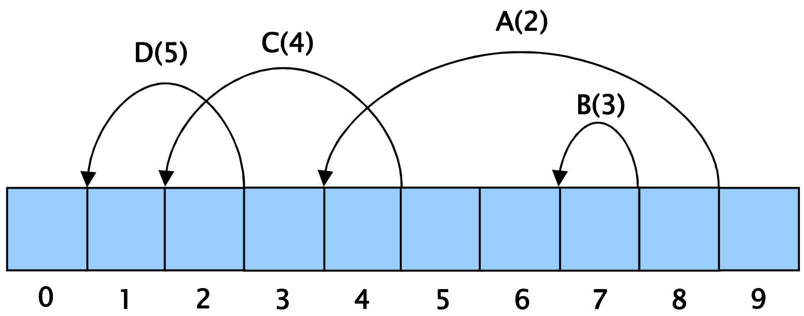


L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)	AB(5)		

Alg:

- foreach cell
  - if L>=length of transition of current column
    - find legal lowest cost multiset of length L, where the transition of current column appears at least once

Select set of transitions



L	A(2)	B(3)	C(4)	D(5)
1		B(3)		
2		BB(6)		D(5)
3		BBB (9)	C(4)	
4		BBBB(12)		DD(10)
5	A(2)	BBBBB(15)	CD(9)	CD(9)
6	AB(5)	AB(5)	CC(8)	DDD(15)

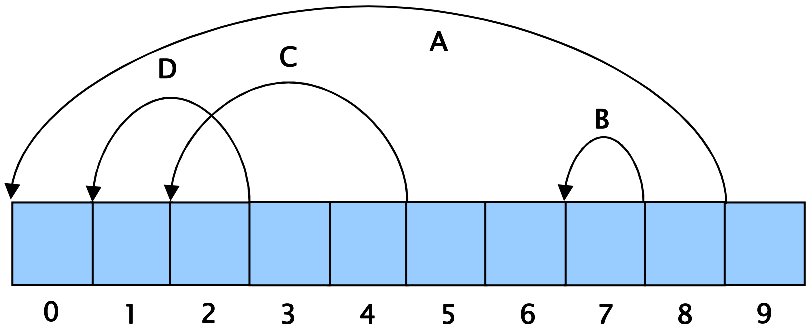
Alg:

- `foreach cell`
  - `if L>=length of transition of current column`
    - `find legal lowest cost multiset of length L, where the transition of current column appears at least once`

- time $O(L^2N^2)$
- space $O(LN)$

- optionally store pointers + range instead of full description

Schedule primitive loops



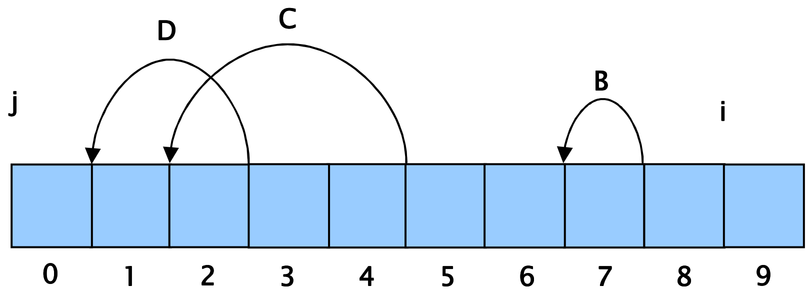
Alg:

Given:

- length =17
- transitions ABCDD occur

current schedule:

Schedule primitive loops



Given:

- length =17
- transitions ABCDD occur

current schedule:

A

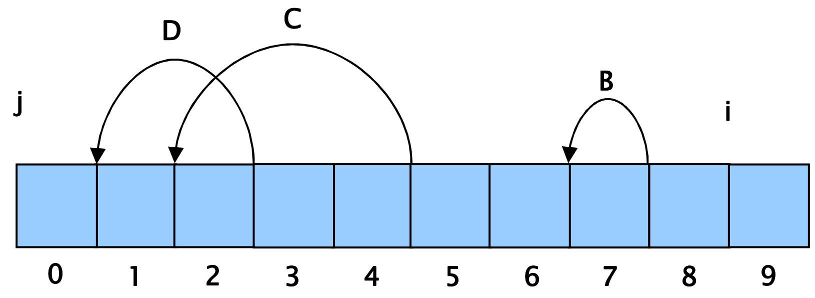
Alg:

- Schedule transition i→j that starts at the end

  → 1 or more ranges

invariant: always in first range

Schedule primitive loops



Given:

- length =17
- transitions ABCDD occur

current schedule:

A D

Alg:

- `Schedule transition i→j that starts at the end`

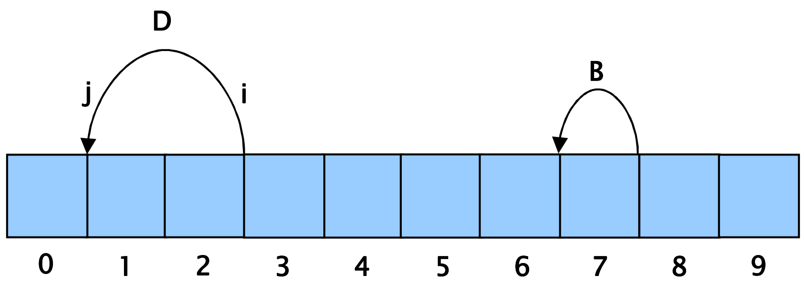
  `→ 1 or more ranges`

- `Schedule any* transition in first range that starts after j`

invariant: always in first range

* deterministic or stochastic

Schedule primitive loops



Given:

- length =17
- transitions ABCDD occur

current schedule:

A D C

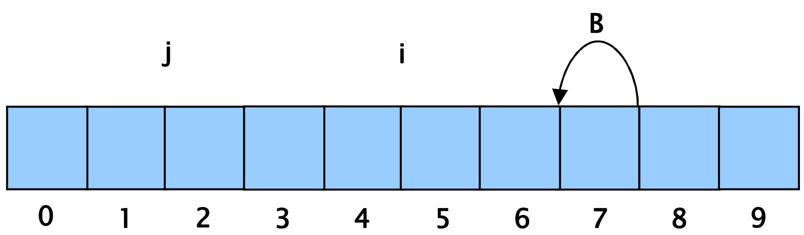
Alg:

- Schedule transition i→j that starts at the end
  → 1 or more ranges
- Schedule any* transition in first range that starts after j

invariant: always in first range

* deterministic or stochastic

Schedule primitive loops



Given:

- length =17
- transitions ABCDD occur

current schedule:

A D C D

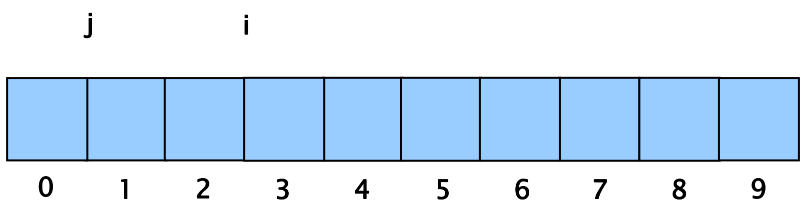
Alg:

- Schedule transition i→j that starts at the end
  → 1 or more ranges
- Schedule any* transition in first range that starts after j

invariant: always in first range

* deterministic or stochastic

Schedule primitive loops



Given:

- length =17
- transitions ABCDD occur

current schedule:

A D C D B

Alg:

- Schedule transition i→j that starts at the end
  → 1 or more ranges
- Schedule any* transition in first range that starts after j

invariant: always in first range

* deterministic or stochastic

Rendering

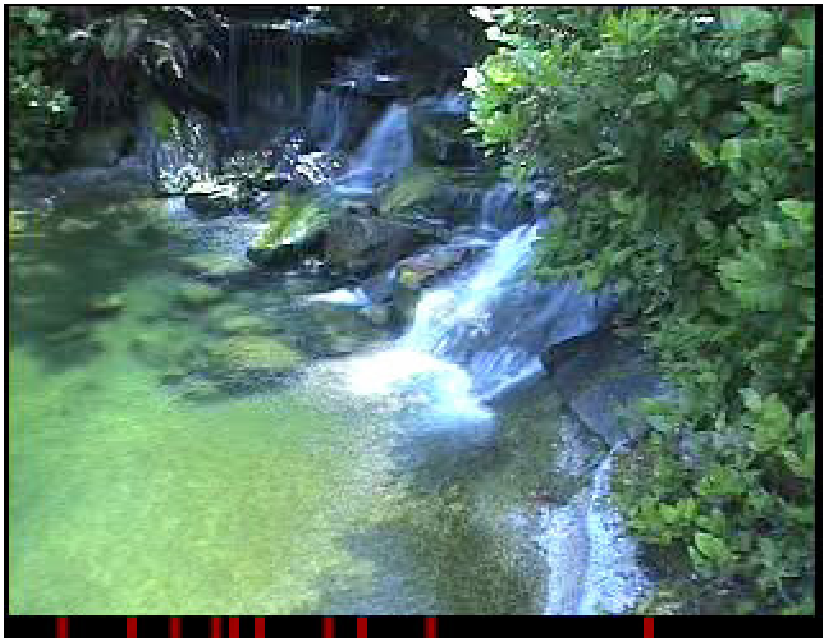
- Jump cut
  - noticable transitions
- Cross-fade
  - blur if misaligned
- Morph
  - common features get aligned

Rendering

flag

Rendering

- Multi-way cross-fade
  - multiple frames displayed at a time
  - weighted average of all participating frames

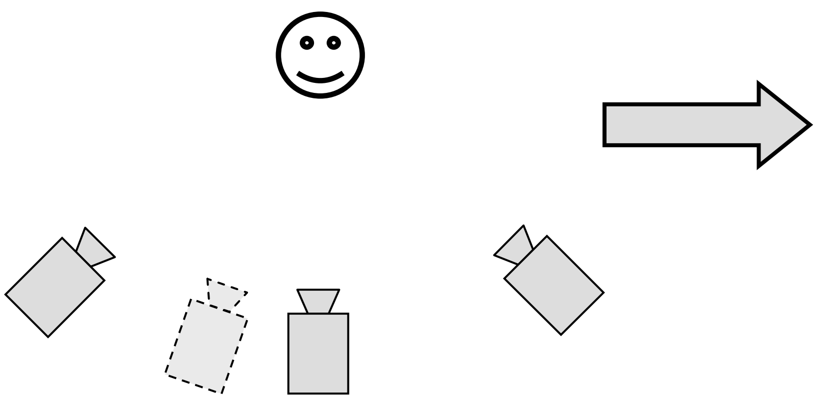


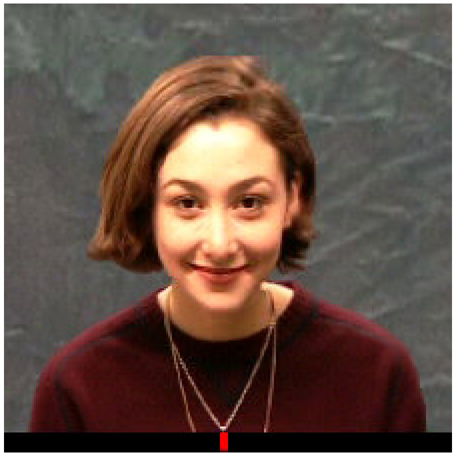
Outline

- Introduction
- Representation
- MAIN SYSTEM
- Extensions
- Results
- Future Work & Discussion

3D Video Texture

- replace video with video texture
- create novel view





Motion factorization

- Divide original video into independently moving parts
  - manual
  - automatic

Video-based Animation

- Control where, how & how long to replay
- Add user-controlled term to error function based on:
  - velocity of video sprite
  - position of frame

Video-based Animation

Video Sprites

1. Object extraction (bg subtraction, blue screen)
2. Store velocity of centroid
3. Center object

→ Analysis + alpha, direction, speed

→ Synthesis + velocities

Extensions

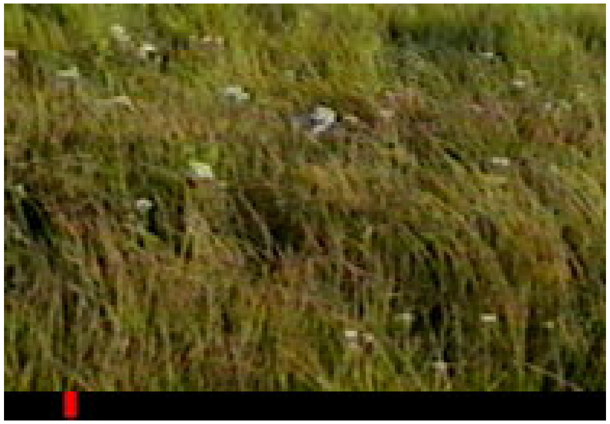
Video Sprites

Results

- Simple algorithm
- Several extensions & applications
- User control
- Combine video textures

Results

- Algorithm fails at complex and high-structured video material



Future Work & Discussion

- Distance metrics
- Blending
- Variety
- Control tools