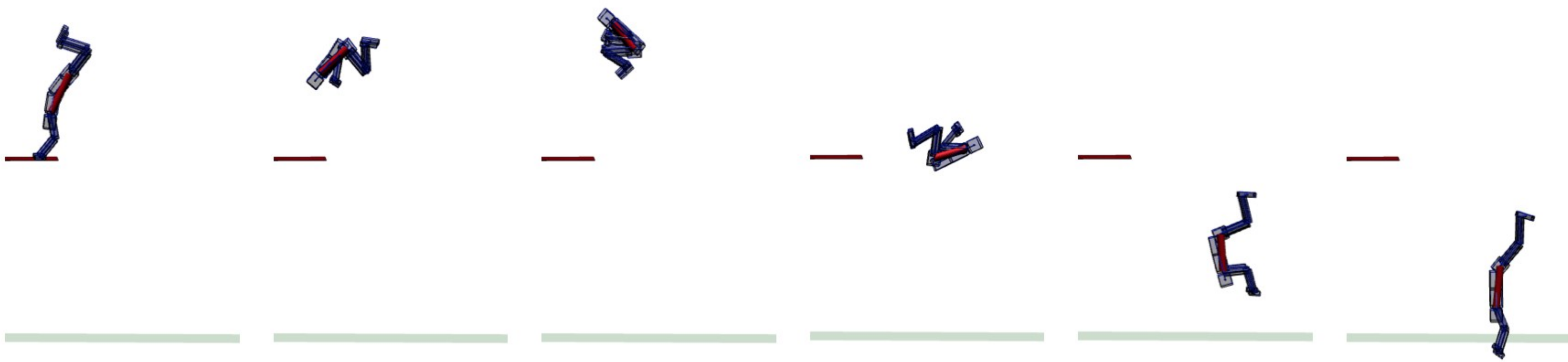
Feedback Control for Rotational Movements in Feature Space

Mazen Al Borno Eugene Fiume Aaron Hertzmann Martin de Lasa

Main problem:

- global orientation
- angular momentum
- inertia

- run computation at interactive rates
- do not rely on any input motion or offline optimization
- time-invariant
- generality



$\mathbf{q}$ - generalized joint positions (vector)

$\mathbf{q}_i$ - controlled variable

$$\left.\begin{aligned} &\mathbf{q}_i \neq \mathbf{q}_i(t) \\ &\mathbf{q}_i = h_i(\theta) \end{aligned}\right\}$$

θ - phase variable
- strictly monotonic with respect to time
- time-invariant controller

$\mathbf{q}$ - generalized joint positions (vector)

$\mathbf{q}_i$ - controlled variable

$\mathbf{q}_i \neq \mathbf{q}_i(t)$

$\mathbf{q}_i = h_i(\theta)$

θ - phase variable
- strictly monotonic with respect to time
- time-invariant controller

Feature space :

- i.e. Center-of-Mass (COM), angular momentum, end-effector
- output of a map of the character's state: $\mathbf{y} = f(\mathbf{q},\mathbf{q'})$ $\mathbf{q'}$ – velocity
- feature-based control

$\mathbf{q}$ - generalized joint positions (vector)

$\mathbf{q}_i$ - controlled variable

$\mathbf{q}_i \neq \mathbf{q}_i(t)$

$\mathbf{q}_i = h_i(\theta)$

θ - phase variable
- strictly monotonic with respect to time
- time-invariant controller

Feature space :
- i.e. Center-of-Mass (COM), angular momentum, end-effector
- output of a map of the character's state: $\mathbf{y} = f(\mathbf{q},\mathbf{q'})$ $\mathbf{q'}$ – velocity
- feature-based control

Let's take it all together : $\mathbf{y}_i = h_i(\theta,\mathbf{q},\mathbf{q'})$ $\mathbf{y}_i$ - desired feature

Note: - In rotational movements θ is an angle in the plane perpendicular to the axis of rotation of the revolution.
- θ increases as the revolution progresses

<u>Benefits of phase variable (parametrizing controllers)</u> :

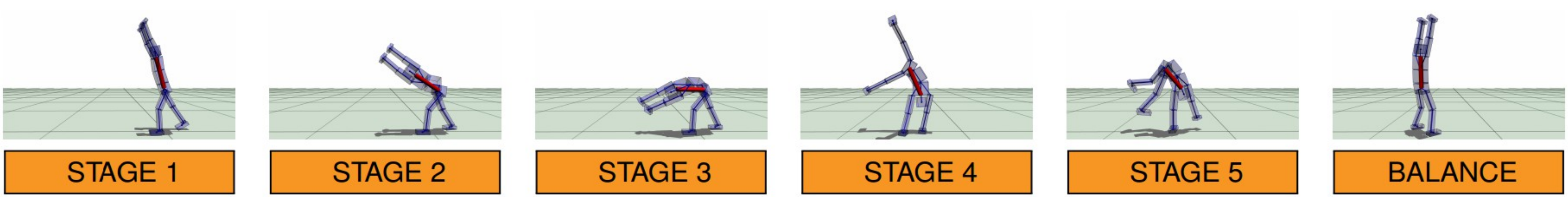
- make the controllers more general
  - i.e. in a flip, the character roughly completes a full revolution, no matter how far, high or fast the character jumps
- make the controllers more robust because of feedback
  - i.e. character performing a cartwheel (no disturbances vs. strong wind forces)

Rotational movements of characters (controllers) :

- Cartwheel
- Flips
- Diving
- Backhandspring
- Pirouette
- Front Aerial
- ...

Rotational movements of characters (controllers) :

- Cartwheel
  - Stage i is triggered when: $\theta \in (l_i, u_i)$
  - Stage 1: $\mathbf{y}_{COM} = \mathbf{c} + \eta\mathbf{d} + [0,l,0]^T$
    - the character tilts towards its left/right side
    - **d** – direction of characters move
    - **c** – position of the COM
    - η and l – scalars: η > 0 and l < 0 (y axis to the ground)
    - active until both hands touch the ground



Rotational movements of characters (controllers) :

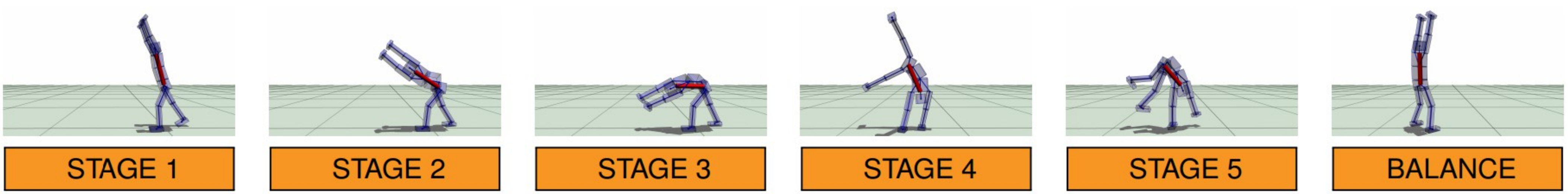
- Cartwheel
  - Stage 2
    - move left character's hand
    - left hand: $y_{lhand}^{xz} = c^{xz} + \eta \cdot d^{xz}$

      $y_{lhand}^{y} = y_{lhand}^{y} \cdot \left(1 - \frac{\theta - \theta_i}{\theta_f - \theta_i}\right)$

      Θ_i - the value of θ when the character should begin to lower its hand

      Θ_f - is the value of θ when the hand should be on the ground
  - Stage 3
    - place the right hand
    - right hand: $y_{rhand}^{y} = y_{rhand}^{y} \cdot \left(1 - \frac{\left|y_{rhand}^{xz} - y_{lhand}^{xz}\right|^2}{s}\right)$

      s – desired distance between the hands when in contact



<u>Rotational movements of characters (controllers)</u> :

- Cartwheel
  - Stage 4
    - hands are deactivated – foots are activated
    - left foot:

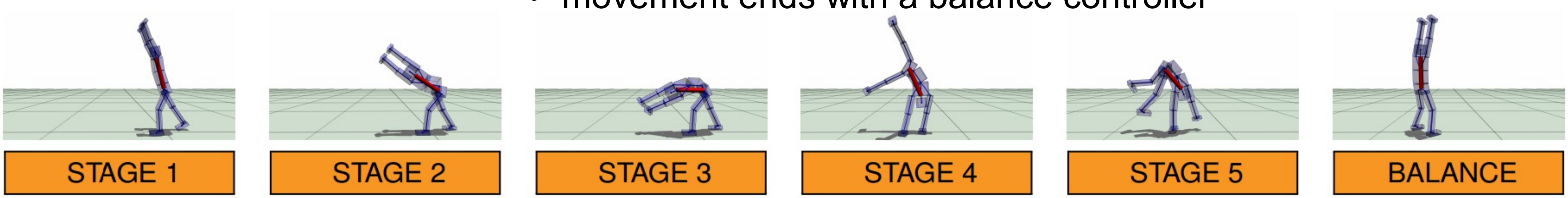
$$y_{lfoot} = R_{\Phi} \cdot (c^{xz} + p^{loff})$$

Φ - is an estimate of the character's global orientation

R_{Φ} - rotation matrix associated with Φ

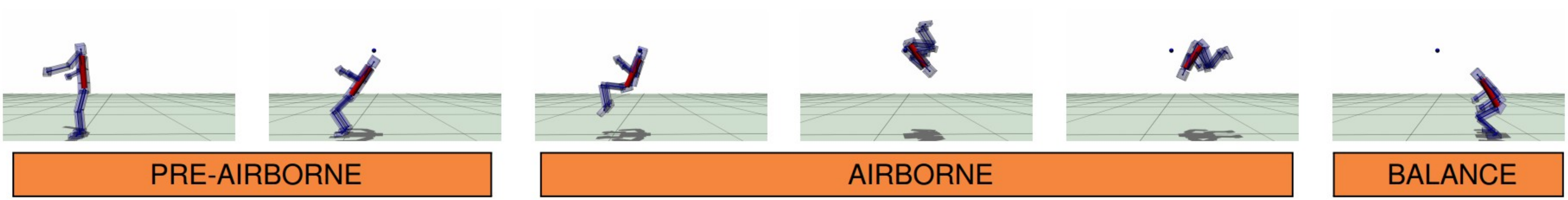
p^{loff} - fixed horizontal offset term for the left foot

    - analogous feedback law applies for right foot
      - orientation of the lumbar joint is a good choice for Φ
  - Stage 5
    - oposite to stage 2 and stage 1 – straighten character
    - movement ends with a balance controller



<u>Rotational movements of characters (controllers)</u> :

- Flips
  - two main stages: pre-airborne & airborne
  - pre-airbone stage:
    - place the character in a crouch position that is slowly tilting on its back
    - generate the necessary angular momentum for the character to flip
    - avoid insufficient or excessive angular momentum



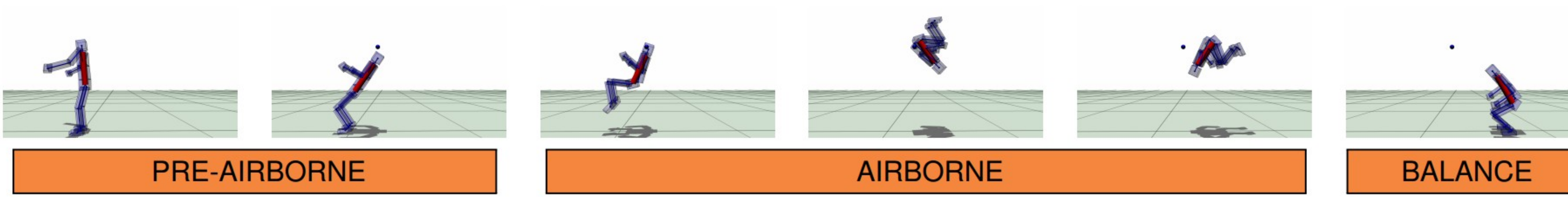
<u>Rotational movements of characters (controllers)</u> :

- Flips
  - two main stages: pre-airborne & airborne
  - pre-airbone stage:
    - place the character in a crouch position that is slowly tilting on its back
    - generate the necessary angular momentum for the character to flip
    - avoid insufficient or excessive angular momentum

estimate the duration of the airborne stage of the flip:

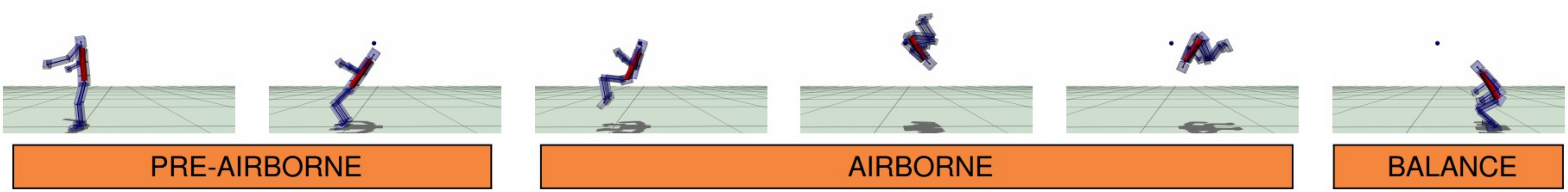
$$t_{air} = \frac{2 \cdot \varepsilon}{g}$$

ε - estimated velocity of the COM
g - gravitational constant



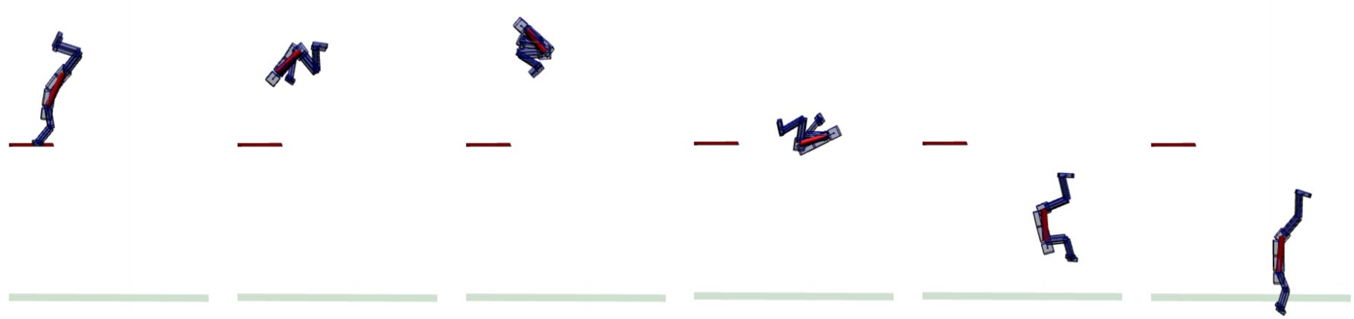
Rotational movements of characters (controllers) :

- Flips
  - two main stages: pre-airborne & airborne
  - airbone stage:
    - land with a desired orientation (θ_{land}) and inertia (I_{land})
    - current angular velocity: $w = \frac{|M|}{I}$ M - angular momentum, I - Inertia
    - desired angular velocity at landing: $w_{land} = \frac{|M|}{I_{land}}$
    - character is rotating too slowly – decrease the character's inertia to increase the angular velocity
    - character is rotating too quickly – increase the character's inertia to decrease the angular velocity



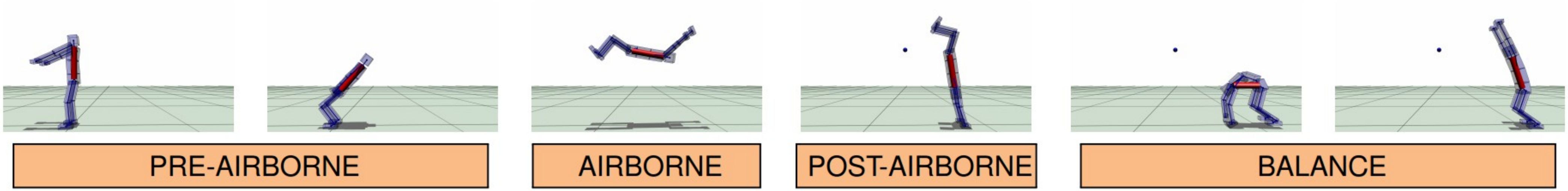
Rotational movements of characters (controllers) :

- Diving
  - The goal is to have the character enter the water with a straight posture and with its arms raised upwards
  - for generality – input can contain desired number of complete revolutions before entry
  - almost identical to flip controller
    - Θ_{land} , I_{land} - different because of the different desired landing positions
    - take into account the number of complete revolutions when calculating the desired average angular velocity
    - shoulders are now included in computation because of arms raising
  - identical airborne controller can generate forward, backward, straight and armstand dives

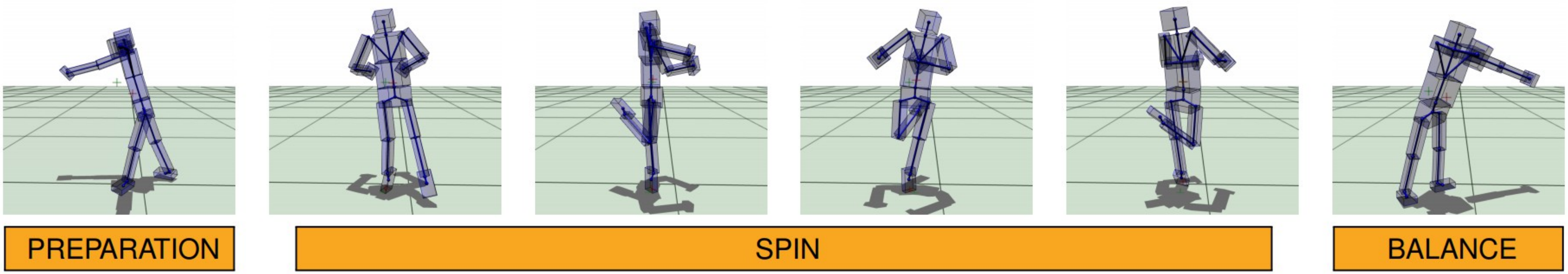


<u>Rotational movements of characters (controllers)</u> :

- Backhandspring
  - pre-airborne stages is different to backflip controllers only in the values of θ_{land} and I_d
  - not significant change of the inertia in the airborne stage
  - rest pose of the shoulders vary when the character is airborne
    - the character raises its arms as a function of θ in order to land on its hands
    - the character enters a post-airborne stage when its hands are close enough to the ground

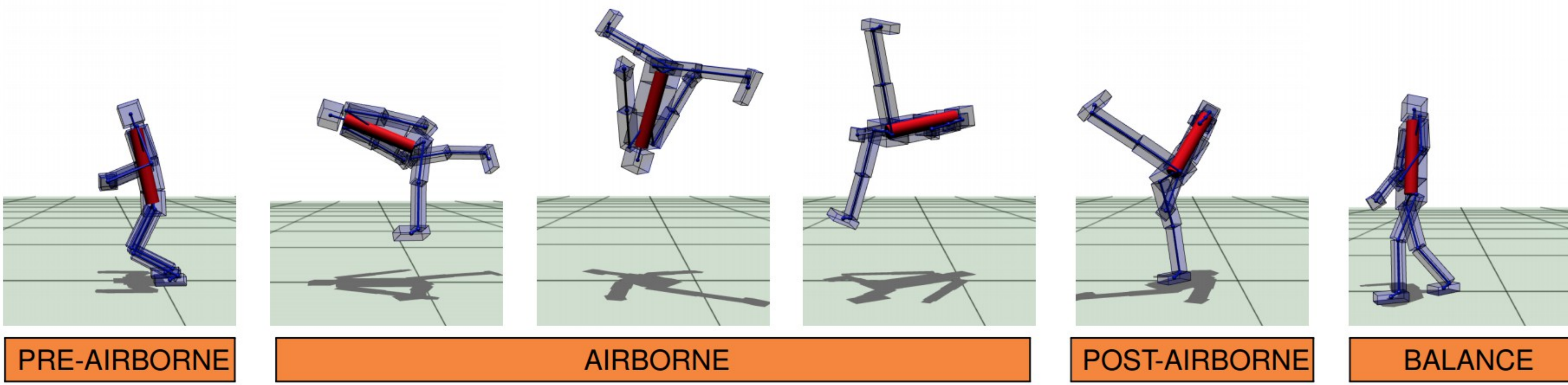


Rotational movements of characters (controllers) :



- Piroutte

- Front Aerial



RESAULTS

- feature-based control algorithms for a wide variety of rotational movements (aerials, cartwheels, dives, and flips), that do not require any input motion or offline optimization
- most of these movements have not previously been generated by physics-based methods without prior data
- controllers are general and robust

- results lack some aspects of natural human motion
- most of the rotations are planar, but phase parametrization can be extended to three-dimensional rotations
- it is not able to synthesize some movements that are at the periphery of performance, such as certain ballet moves that require an enormous amount of precision

Thank you for your attention

Peter Kovács

Ján Vodila