Exercise Multimedia Technology WS 2003/2004

Sheet 3 (November 7th 2003)

3.1 Single image compression

Let's consider two types of images, one being taken from the real world ("Fotos"), the other being computer generated like e. g., cliparts.

- Discuss the compression methods JPEG and GIF with regard to their usefulness for the two types of images. How could preprocessing the image improve the compression ratio?
- How does JPEG work? What kinds of images are especially suitable for JPEG compression? How is it possible to improve the compression ratio, in this case by applying preprocessing?

JPEG is a lossy kind of image compression that is based on the discrete cosine transformation, whereas GIF is a lossless type based on Lempel-Ziv-Welch compression. GIF compression works best for images with long runs of the same color or with repeating pixel patterns. This is typical for computer generated images like cliparts or cartoons. Reducing the amount of colors used in an image by quantizing them would also increase the likelihood of longer continuous runs.
A simple and straightforward solution to improve the compression ratio for JPEG would be to filter the image using a low pass filter (e. g. using a gaussian filter) before compressing it. Low-pass filtering discards high frequencies (or wave numbers) from the image (this answer would be sufficient).
However, note that the DCT itself already is a technique for decomposing an image into its contained frequencies. So discarding those coefficients that are responsible for the higher frequencies would also do the job. Rather than doing the filtering in the spatial domain, it would make even more sense to filter in the frequency domain by discarding coefficients.

3.2 Block truncation coding

A tiny image is defined by the following table.

92	108
106	94

Code the image using the block truncation coding (BTC) algorithm.

$$\mu = \frac{92+108+106+94}{4} = 100$$

$$\sigma = \sqrt{\frac{(100-92)^2+(100-108)^2+(100-106)^2+(100-94)^2}{4}} =$$

$$\sqrt{\frac{64+64+36+36}{4}} = \sqrt{200/4} = \sqrt{50}$$

Bitstring to be transmitted:

1	0
0	1

$$a = 100 - \sqrt{50}\sqrt{\frac{2}{2}} \approx 93$$

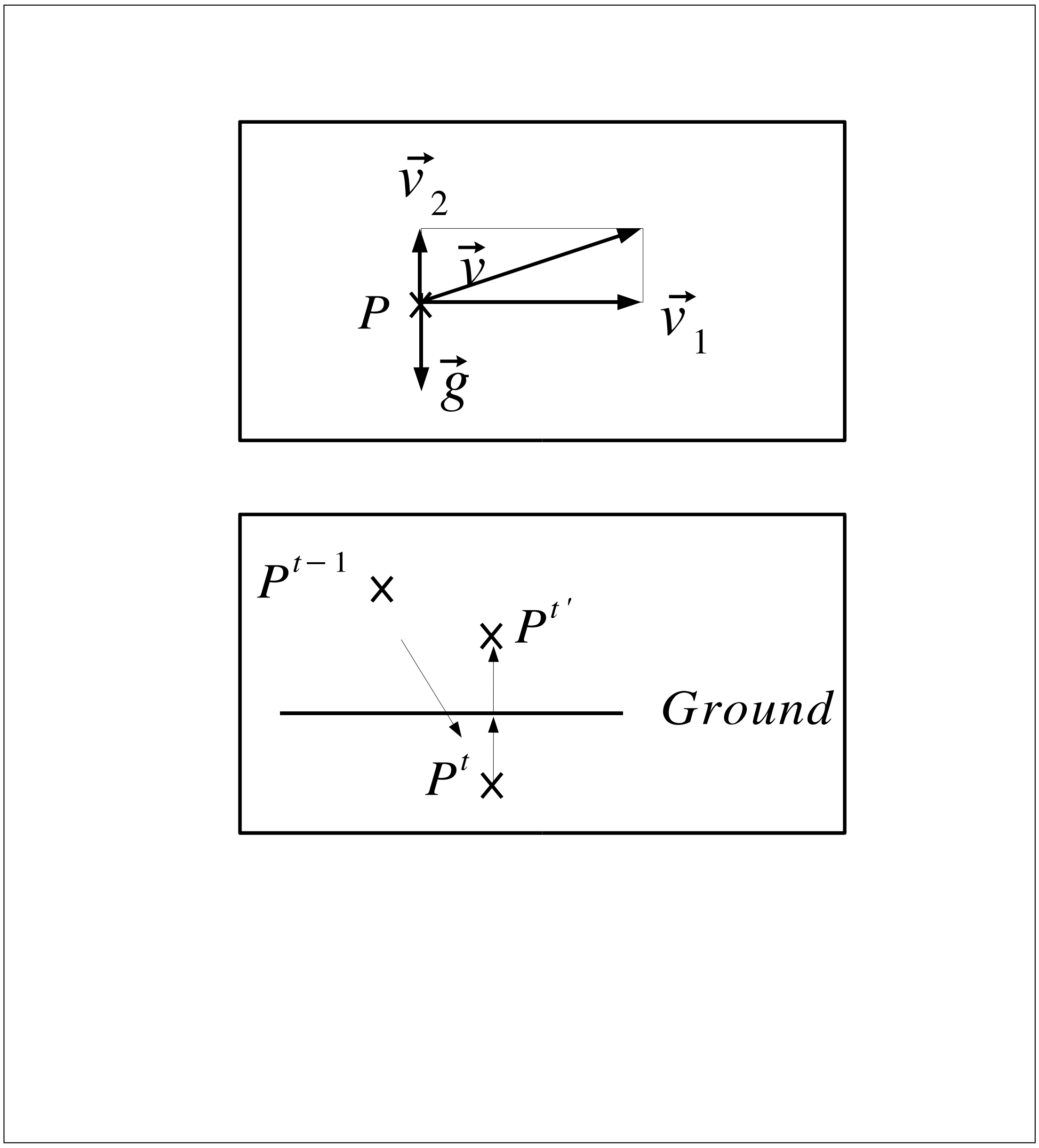
$$b = 100 + \sqrt{50}\sqrt{\frac{2}{2}} \approx 107$$

3.3 Animation

We have learned in the lecture how articulated structures can help the animator to create realistic motions of a skeleton. The idea is to limit the degrees of freedom in order to allow only valid motion. Even more realism could be achieved by modelling motion according to the laws of physics.

Simulate the motion of a jumping ball yourself. The user should be able to place the ball initially somewhere on the screen. At the beginning it should be assigned an initial random motion. It should then fall to the ground and bounce back.

Hint: The motion (vector) of the ball consists of a vertical and a horizontal component. In each time unit gravity “pulls” on the vertical component. The horizontal component is never influenced. The motion vector is used to update the location of the ball between two frames of the animation. After bouncing off the ground some kinetic energy should be lost.



Each particle is associated with a location P and its current direction $\vec{v}$. The particle's location is simply updated by

$$P_{t+1} = P_t + \vec{v}$$

At the same time the vertical component v_2 is incremented by a gravity constant g because the gravity pulls at the particle in each time unit. The horizontal component is never influenced (except if you want to model the influence of the air resistance).

$$v_1 = v_1 + g$$

Finally, how the particle bounces also has to be simulated. This is obtained simply by mirroring the particle with the ground as the mirror axis once the particle is below the ground. At the same time, the component v_2 has to be multiplied by -1 so that the particle changes its direction. In order to ensure that energy is lost at collision a value slightly larger than -1 can be used. Once again the bounce does not influence the horizontal component v_1.
Sample solution:
www.informatik.uni-mannheim.de/pi4/data/simulation.tgz