Musculoskeletal imaging, analysis and model building in clinical reasoning 2 (MIAMI-2)

Academic year 2023-2024

Student's module book

Please think green, consider the environment before printing

Content

Inhoud

Part 1 General information

1 Planning group and coordination

You will receive this module during the second semester. The module covers 12 ECTs. This module focusses on musculoskeletal complaints, diagnostic motion analysis, the related biomedical engineering and the analysis of the results. During the intensive on campus training week there will be sessions in function of measuring movement, medical imaging, clinical applications on the VUB campus supplemented with visits of clinics or centers where the use of technology is a priority.

This module is organized using a blended learning approach. Lectures will be online (asynchronous (= pre-recorded) and synchronous (= live)). Working sessions, responsory colleges and Q&A's will be live online. Next to this, you will also work on a portfolio.

At the end of the module during the intensive week, you will be evaluated.

The module has been developed by a group of teachers with different backgrounds. For general questions about the module, please contact the module coordinator. You will also notice that, because of the size of the module and the large amount of topics, several guest lectures will be involved. Due to this, the module book will consist of a set of PowerPoint presentations rather than an extensive textbook.

Role	**Name**	**e-mail**	**Picture**
Module coordinator / Tutor clinical reasoning	Matthias Eggermont	matthias.eggermont@vub.be	
Tutor movement registration, biomechanics, clinical applications & AI	Bart Jansen	bart.jansen@vub.be	
Tutor applied biomechanics & clinical applications	Eva Swinnen	eva.swinnen@vub.be	
Tutor movement registration	Benyameen Keelson	benyameen.keelson@vub.be	
Tutor movement registration	Erik Cattrysse	erik.cattrysse@vub.be	
Tutor clinical applications / Tutor clinical reasoning	David Beckwée	david.beckwée@vub.be	

2 Introduction

The main objective of this program is to educate a new type of health professional that combines competencies in regard to prevention, vitality and rehabilitation in primary care (physiotherapy) with competencies in medical engineering, ICT and technology.

Students in this program will learn to invent and think of technological solutions for challenges in prevention, vitality and rehabilitation to enhance personalized patient care. In the end, this will lead to the prevention or delay of functional decline and more effective rehabilitation strategies for patients.

The students will also gain transversal skills, which allow them to form a bridge between health care (and it's specific challenges), technology and business/industry, and to manage and communicate their projects in the digital era.

2.1 Content and aim of the module

2.1.1 Content

In this course there will be 5 major parts:

- An introductory path
  Basic information about the module and specific background knowledge (via knowledge clips, papers, …) will be shared via the learning platform. This will be a kind of library where students can look-up necessary knowledge when it becomes relevant (just in time learning).

- Movement registration
  This topic consists of lectures and working sessions about sensors and technologies used in movement analysis. Some of the tools that are subject of this topic are: electromagnetic trackers, accelerometers, IMU sensors, marker and markerless motion capture, EMG, forceplates … . Dynamic imaging techniques are also subject of this topic. At the end of the module, during the intensive week, an on-campus laboratory visit is planned to get familiar with and work with different kind of techniques.

- Advanced biomechanics & clinical analysis
  This topic consists of lectures and working sessions about kinematic and kinetic analysis to support the clinical analysis of 3D imaging techniques. Kinematic data include displacement and orientation of body segments, joint angles and spatio-temporal gait parameters. Kinetic data include ground reaction forces, lower limb joint mechanical moments and powers, kinetic and potential energy. Muscle activation patterns are analyzed through the electrical signals (EMG) associated to muscular fiber contraction. Therefore, also the analysis of EMG data is part of this topic. Clinical reasoning will be also part of this subject when talking about healthy vs pathologic patterns in the part of applied biomechanics. At the end of the module, during the intensive week, an on-campus laboratory visit is planned in the laboratory.

- Clinical Applications, including remote rehabilitation
  This topic consists of lectures and working sessions about applications/technologies that can be used for diagnostic and/or therapeutic purposes designed for clinical use, but also in the field of telerehabilitation and telemonitoring (remote). Some of the subjects: mobile health (mHealth) technology and wearables, serious games, robotics, AR/VR, …. Clinical reasoning will be also part of this subject when making decisions about usability of these applications/technologies. At

the end of the module, during the intensive week, an on- campus visit is planned to work with these clinical applications. Clinical reasoning will be trained in more complex patient case-studies and more complex methodologies will be applied.

- Artificial Intelligence
  This topic consists of lectures and working sessions on how the basics of artificial intelligence and machine learning. After an introduction on the main paradigms of (un)supervised learning and reinforcement learning, the focus is mainly on supervised learning methods, illustrated for instance by means of decision trees, random forests and neural networks. Also rule based systems are introduced in the context of expert systems. Natural language processing is introduced as an interface for such systems. Deep learning is introduced as an extension of neural networks, primarily focusing on image processing tasks. Applications of all techniques in the domain of rehabilitation engineering are provided extensively.

2.1.2 Learning outcomes

1. The student has a profound knowledge and understanding of how to measure movement using a broad spectrum of sensors.
2. The student has a profound knowledge and understanding of advanced (dynamic) imaging techniques.
3. The student has profound understanding of and is able to use Python programming language.
4. The student has profound knowledge in kinematic analysis.
5. The student is able to interpret and analyze the applicability of (arthro)kinematic evidence and knowledge in relation to the clinical analysis.
6. The student has profound knowledge in kinetic analysis.
7. The student is able to apply knowledge about biomechanics in a medical context.
8. The student has insight in clinical and remote applications to diagnose, monitor and train patients or healthy people.
9. The student is aware of current trends in Artificial Intelligence and has the competence to judge different approaches and technologies.
10. The student has theoretical and practical knowledge of the basic concepts of supervised and unsupervised learning and reinforcement learning.
11. The student can enumerate strengths and weaknesses of different AI technologies and under what circumstances they can be applied.
12. The student should be able to apply machine learning techniques and to tune the parameters of the chosen algorithm.
13. The student should be able to communicate with experts about machine learning problems.
14. The student has insight into which patients can benefit from machine learning techniques and how to apply these techniques to the clinical problem. The student also has insight in methodological issues involved.
15. The student not only has theoretic knowledge of the topics under consideration but also practical hands-on experience.
16. The student will also learn about open research questions to stimulate their own explorations in the field.
17. The use of the Python ecosystem should enable the student to write programs to solve problems.
18. The student is able to incorporate all obtained knowledge into their clinical reasoning for diagnostic, therapeutic purposes or within prognostic prevention.
19. The student should also be able to report and to present the results of his or her experiments to both specialists and non-specialists.

3 Educational activities

This module will consist of theoretical lessons, practical sessions, portfolio and an intensive on-campus week at the Vrije Universiteit Brussel. For more information about the planning of the lessons, if the lessons are synchronous (live, online), asynchronous (pre-recorded) or for the planning of the on-campus week, check the information on the digital learning platform.

For questions about the educational activities or planning, please contact the module coordinator.

4 Testing and grading (evaluation)

To evaluate the different topics in this module we use a portfolio to check your progress. We will also take into account your progression during online working sessions.

The **final grade** is based on the following rubric:

- **Fail**:
  - The student did not meet at least the satisfactory level.

- **Satisfactory**:
  - The student has basic knowledge and understanding to use Python programming language on exercises on movement registration, data (biomechanics, clinical analysis) manipulation and in the light of AI, on decision trees, random forests and neural networks.
  - The student is able to link the newly acquired knowledge with patient cases and daily practice.
  - The student is capable of communicating interdisciplinary with oa. engineers about all steps that were necessary in the exercises, hurdles, defects and possibilities in an acceptable manner.
  - The student was able to finalize all basic exercises.

- **Good**:
  - The student has good knowledge and understanding to use Python programming language on movement registration, data (biomechanics, clinical analysis) manipulation and in the light of AI, on decision trees, random forests and neural networks.
  - The student is able to link the newly acquired knowledge with patient cases and daily practice.
  - The student is capable of communicating interdisciplinary with oa. engineers about all steps that were necessary in the exercises, hurdles, defects and possibilities in a professional manner.
  - The student was able to finalize more advanced exercises.

- **Excellent**:
  - The student has excellent knowledge and understanding to use Python programming language on movement registration, data (biomechanics, clinical analysis) manipulation and in the light of AI, on decision trees, random forests and neural networks.
  - The student is able to link the newly acquired knowledge with patient cases and daily practice and already started to use the newly acquired skills in practice.

- The student is capable of communicating interdisciplinary with oa. engineers about all steps that were necessary in the exercises, hurdles, defects and possibilities in a skilled manner.
- The student was able to finalize all exercises.

For more detailed information about the grading, check the digital learning platform or ask the dedicated teaching staff of each topic.

5 Literature

Literature is described in each separate section in this module book. Mainly the references can be retrieved from PubMed or via the VUB library.

6 Sessions

See the online planning for up-to-date information about the lessons.

Part 2 Introductory path

1 Introduction to mathematics

Who am I:

- Prof. Dr. Bart Jansen, bjansen@etrovub.be
- Prof at Department of Electronics and Informatics (ETRO) of the Vrije Universiteit Brussel (VUB)
- PhD in Computer Science, Artificial Intelligence
- Coordinating research on signal processing and AI in medical applications.
- A lot of interest in Rehabilitation Engineering

Objectives:

- To learn the basics of the Python language.
- To learn to express simple ideas in Python.
- To refresh some rusty mathematical concepts.
- Next semester, we will need all three ...

Expected efforts:

- You program as much as ever possible.
- Programming is easy once you can do it ;-)
- There is only one way and it is the hard way.
- Get a python environment today!
  (Pycharm, Anaconda, Canopy, Wing, Eclipse ...)

How:

- online synchronous (live) (see online schedule)
- followed by online (live) working sessions (see online schedule)
- and on campus working sessions (see online schedule intensive week)

1.1 Chapter 1: Introduction to mathematics

: Chapter 1

1.1.1 Types of research

1.1.2 Linear equations

1.1.3 Set of linear equations

1.1.4 Trigonometric functions

1.2 Additional information, slides, articles

CHAPTER 1

TYPES OF RESEARCH
QUALITATIVE VS QUANTITATIVE

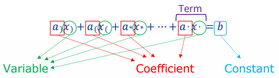
Qualitative research: e.g., investigate the quality of a movement (normal or abnormal)

Quantitative research: Describe the movement/a physical state with physical quantities and a mathematical model.

Example of a quantitative model: The friction due to air during cycling is related to the velocity. If the velocity doubles, the friction increases with a factor 4.

$$F_{friction} = -kv^2$$

LINEAR EQUATIONS
DEFINITION



Properties:

1. The exponent of the variables is equal to 1 (first order equation)
2. A term is composed of one variable and one coefficient

LINEAR EQUATIONS
EXAMPLES

Examples of linear equations:

12x-1=3

3x+5y=17

$-10F_1+8F_2+4F_3=100$

3p+4q-7r=12

Examples of equations that are not linear:

$3a^4+6b=18$

xy+xz=8

$3e^{-4x}+7y=0$

LINEAR EQUATIONS
HOW TO SOLVE A LINEAR EQUATION

Rules:

1. You can add/subtract the same number from the left and right side of the equation.

$$12x - 1 = 3 \overset{+1}{\Leftrightarrow} 12x = 4$$

2. You can multiple the left and right side of the equation by the same number.

$$12x = 4 \overset{\cdot\frac{1}{4}}{\Leftrightarrow} 3x = 1$$

SET OF LINEAR EQUATIONS
DEFINITION

$$\begin{cases} a_1x_1 + a_2x_2 + a_3x_3 = a_4 \\ b_1x_1 + b_2x_2 + b_3x_3 = b_4 \\ c_1x_1 + c_2x_2 + c_3x_3 = c_4 \end{cases}$$

Properties:

1. Every equation is linear
2. A set of linear equations can be solved if the number of unknown variables is equal to the number of independent equations

Example:

$$\begin{cases} 2x_1 + 3x_2 = 3 \\ 8x_1 - 2x_2 = 26 \end{cases}$$

SET OF LINEAR EQUATIONS
HOW TO SOLVE A SET OF LINEAR EQUATIONS

Method 1: Substitution method

$$\begin{cases} 2x_1 + 3x_2 = 3 \ (1) \\ 8x_1 - 2x_2 = 26 \ (2) \end{cases}$$

Step 1: Solve equation (1) for x_1

$$x_1 = \frac{3 - 3x_2}{2} = 1.5 - 1.5x_2$$

Step 2: Substitute x_1 in equation (2)

$$\begin{cases} x_1 = 1.5 - 1.5x_2 \ (3) \\ 8(1.5 - 1.5x_2) - 2x_2 = 26 \ (4) \end{cases}$$

Step 3: Solve equation (4) for x_2

$$8(1.5 - 1.5x_2) - 2x_2 = 26$$
$$\Leftrightarrow -14x_2 = 14$$
$$\Leftrightarrow x_2 = -1$$

Step 3: Substitute x_2 in equation (1/3)

$$\begin{cases} x_1 = 3 \\ x_2 = -1 \end{cases}$$

SET OF LINEAR EQUATIONS
HOW TO SOLVE A SET OF LINEAR EQUATIONS

Method 1: Elimination method

$$\begin{cases} 2x_1 + 3x_2 = 3 \ (1) \\ 8x_1 - 2x_2 = 26 \ (2) \end{cases}$$

Step 1: Eliminate x_1 from equation 2:

$$\begin{cases} 2x_1 + 3x_2 = 3 \Leftrightarrow 8x_1 + 12x_2 = 12 \\ 8x_1 - 2x_2 = 26 \Leftrightarrow -8x_1 + 2x_2 = -26 \end{cases}$$

Multiply the equations

$$\begin{array}{r} 8x_1 + 12x_2 = 12 \\ -8x_1 + 2x_2 = -26 \\ \hline 0x_1 + 14x_2 = -14 \end{array}$$

Add the equations

The set of equations becomes:

$$\begin{cases} 2x_1 + 3x_2 = 3 \ (3) \\ -14x_2 = 14 \ (4) \end{cases}$$

Step 2: Solve equation (4) for x_2

$$-14x_2 = 14 \Leftrightarrow x_2 = -1$$

Step 3: Substitute x_2 in equation (3), and solve for x_1

$$2x_1 + 3 \cdot -1 = 3$$
$$x_1 = 3$$

SET OF LINEAR EQUATIONS
EXAMPLE 1: ELIMINATION METHOD

$$\begin{cases} 4x + y = 14.5 \ (1) \\ 3x + 2y = 16.5 \ (2) \end{cases}$$

Eliminate x from equation (1):

$$\begin{cases} 4x + y = 14.5 \Leftrightarrow 12x + 3y = 43.5 \\ 3x + 2y = 16.5 \Leftrightarrow -12x - 8y = -66 \end{cases}$$

$$\begin{array}{r} 12x + 3y = 43.5 \\ -12x - 8y = -66 \\ \hline -5y = -22.5 \end{array}$$

Equation (1) becomes $-5y = -22.5$

$$\begin{cases} -5y = -22.5 \ (1) \\ 3x + 2y = 16.5 \ (2) \end{cases}$$

Solve equation (1):

$$-5y = -22.5 \Leftrightarrow y = 4.5$$

Solve equation (2):

$$3x + 2y = 16.5$$
$$3x + 2 \cdot 4.5 = 16.5$$
$$x = 2.5$$

SET OF LINEAR EQUATIONS
EXAMPLE 2: ELIMINATION METHOD

$$\begin{cases} 3x + 2y - z = 4 \ (1) \\ x + y + z = 6 \ (2) \\ 2x - 2y + 3z = 7 \ (3) \end{cases}$$

Eliminate x from equation (1):

$$\begin{cases} 3x + 2y - z = 4 \Leftrightarrow -3x - 2y + z = -4 \\ x + y + z = 6 \Leftrightarrow 3x + 3y + 3z = 18 \end{cases}$$

$$\begin{array}{r} -3x - 2y + z = -4 \\ 3x + 3y + 3z = 18 \\ \hline y + 4z = 14 \end{array}$$

Equation (1) becomes $y + 4z = 14$

Eliminate x from equation (2):

$$\begin{cases} 2x - 2y + 3z = 7 \Leftrightarrow -2x + 2y - 3z = -7 \\ x + y + z = 6 \Leftrightarrow 2x + 2y + 2z = 12 \end{cases}$$

$$\begin{array}{r} -2x + 2y - 3z = -7 \\ 2x + 2y + 2z = 12 \\ \hline 4y - z = 5 \end{array}$$

The set of equations becomes:

$$\begin{cases} y + 4z = 14 \ (4) \\ x + y + z = 6 \ (5) \\ 4y - z = 5 \ (6) \end{cases}$$

SET OF LINEAR EQUATIONS
EXAMPLE 2: ELIMINATION METHOD

$$\begin{cases} y + 4z = 14 \ (4) \\ x + y + z = 6 \ (5) \\ 4y - z = 5 \ (6) \end{cases}$$

Eliminate z from equation (6):

$$\begin{cases} y + 4z = 14 \Leftrightarrow y + 4z = 14 \\ 4y - z = 5 \Leftrightarrow 16y - 4z = 20 \end{cases}$$

$$\begin{array}{r} y + 4z = 14 \\ 16y - 4z = 20 \\ \hline 17y = 34 \end{array}$$

$$\Leftrightarrow y = 2$$

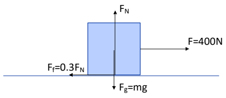
Solve equation (4):

$$y + 4z = 14$$
$$\Leftrightarrow 2 + 4z = 14$$
$$\Leftrightarrow z = 3$$

Solve equation (5):

$$x + y + z = 6$$
$$\Leftrightarrow x + 2 + 3 = 6$$
$$\Leftrightarrow x = 1$$

SET OF LINEAR EQUATIONS
EXAMPLE 3



A box with a mass of 50 kg is pulled horizontally over the ground with a force F equal to 400N.

We know from the 2nd law of Newton:

$$\begin{cases} F_0 = ma_0 \\ F_1 = ma_1 \end{cases} \Rightarrow \begin{cases} F - F_2 = ma_0 \\ F_3 - F_4 = 0 \end{cases} \Rightarrow \begin{cases} F - 0.3 \cdot F = ma_0 \\ F_3 - mg = 0 \end{cases} \Rightarrow \begin{cases} 400N - 0.3 \cdot F_3 = 50kg \cdot a_0 \\ F_3 - 50kg \cdot 10\frac{N}{kg} = 0 \end{cases}$$

$$\Rightarrow \begin{cases} a_5 = \dfrac{400N - 0.3 \cdot F_3}{50kg} \\ F_3 = 500N \end{cases} \Rightarrow \begin{cases} a_5 = 5\dfrac{m}{s^2} \\ F_3 = 500N \end{cases}$$

SET OF LINEAR EQUATIONS
EXAMPLE 4

Imagine, when you run for 1 hour and do 10 push-ups, then you need to drink 1 liter of water. When you run for 2 hours and do 20 push-ups, you drink 2 liters of water. How much water would you drink if you would run only 1 hour. And how much water would you need to drink per push-up?

$$\begin{cases} 1r + 10p = 1 \\ 2r + 20p = 2 \end{cases}$$

Elimination method:

$$\begin{cases} 1r + 10p = 1 \\ 2r + 20p = 2 \end{cases} \Rightarrow \begin{cases} 2r + 20p = 2 \\ 2r + 20p = 2 \end{cases} \Rightarrow \begin{cases} 2r + 20p = 1 \\ 0 = 0 \end{cases}$$

The equations are not independent! An infinite number of solutions exist.

SET OF LINEAR EQUATIONS
EXAMPLE 5

Imagine, when you run for 1 hour and do 10 push-ups, then you need to drink 1 liter of water. When you run for 2 hours and do 20 push-ups, you drink 3 liters of water. How much water would you drink if you would run only 1 hour. And how much water would you need to drink per push-up?

$$\begin{cases} 1r + 10p = 1 \\ 2r + 20p = 3 \end{cases}$$

Elimination method:

$$\begin{cases} 1r + 10p = 1 \\ 2r + 20p = 3 \end{cases} \Rightarrow \begin{cases} 2r + 20p = 2 \\ 2r + 20p = 3 \end{cases} \Rightarrow \begin{cases} 2r + 20p = 1 \\ 0 = 1 \end{cases}$$

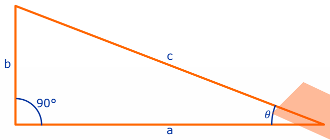
0=1
This is not possible. So, no solution exists.

TRIGONOMETRIC FUNCTIONS
DEFINITION

Sine: $\sin(\theta) = \frac{6}{\$}$

Cosine: $\cos(\theta) = \frac{7}{\$}$

Tangent: $\tan(\theta) = \frac{6}{7}$

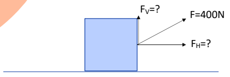


The angle θ is expressed in degrees (°) or radians (rad).

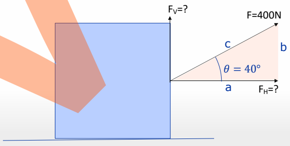
$1° = \frac{(8}{*9:} rad = 0.0175\, rad$

$1 rad = \frac{*9:°}{(8} = 57.3°$

TRIGONOMETRIC FUNCTIONS
EXAMPLE



A box is pulled over the ground with a force F equal to 400N. The angle between the force and the ground is equal to 40°. What is the horizontal force (F_H) and the vertical force (F_V)?



$\cos(\theta) = \frac{a}{c}$

$\cos(\theta) = \frac{F_<}{F}$

$F_< = \cos(\theta) \cdot F$

$F_< = \cos(40°) \cdot 400N$

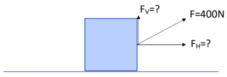
$\sin(\theta) = \frac{b}{c}$

$\sin(\theta) = \frac{F_=}{F}$

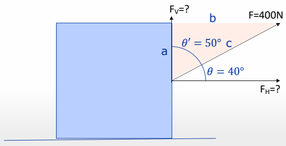
$F_= = \sin(\theta) \cdot F$

$F_= = \sin(40°) \cdot 400N$

TRIGONOMETRIC FUNCTIONS
EXAMPLE



A box is pulled over the ground with a force F equal to 400N. The angle between the force and the ground is equal to 40°. What is the horizontal force (F_H) and the vertical force (F_V)?



$\cos(\theta') = \frac{a}{c}$

$\cos(\theta') = \frac{F_=}{F}$

$F_= = \cos(\theta') \cdot F$

$F_= = \cos(50°) \cdot 400N$

TRIGONOMETRIC FUNCTIONS
UNIT CIRCLE

Sine: $\sin(\theta) = \frac{6}{\$} = \frac{6}{)} = b$

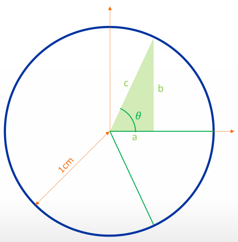
Cosine: $\cos(\theta) = \frac{7}{\$} = \frac{7}{)} = a$

Tangent: $\tan(\theta) = \frac{6}{7}$

$-1 \leq \sin(\theta) \leq 1$

$-1 \leq \cos(\theta) \leq 1$

$\sin(\theta)^(+ \cos(\theta)^(= 1$



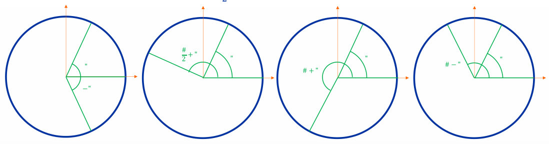
TRIGONOMETRIC FUNCTIONS
RULES

cos($) = cos(−$) sin($) = −sin(−$)

sin($) = −./0(½ + $) cos($) = 012(½ + $)

sin($) = −sin(, + $) cos($) = −cos(, + $)

sin($) = sin(, − $) cos($) = −cos(, − $)

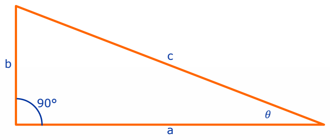


TRIGONOMETRIC FUNCTIONS
INVERSE FUNCTIONS

Arcsine: $\sin\left(\frac{6}{5}\right)^{-1} = \theta$

Arccosine: $\cos\left(\frac{7}{5}\right)^{-1} = \theta$

Arctangent: $\tan\left(\frac{6}{7}\right)^{-1} = \theta$

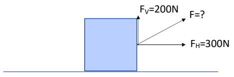


The angle θ is expressed in degrees (°) or radians (rad).

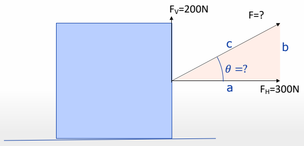
$1° = \frac{(8}{*9:} rad = 0.0175\, rad$

$1 rad = \frac{*9:°}{(8} = 57.3°$

TRIGONOMETRIC FUNCTIONS
EXAMPLE



A box is pulled over the ground. The horizontal force F_H and the vertical force F_V are equal to 300N and 200N. What is the angle between the resultant force F and the ground?



$\tan\left(\frac{b}{a}\right)^{-1} = \theta$

$\tan\left(\frac{F_=}{F_<}\right)^{-1} = \theta$

$\tan\left(\frac{200}{300}\right)^{-1} = \theta$

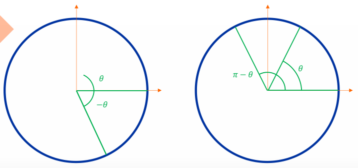
TRIGONOMETRIC FUNCTIONS
INVERSE FUNCTIONS ON THE UNIT CIRCLE

Multiple solutions exist:

$\cos(x)^{-1} = \theta + k \cdot 2\pi \; or \; -\theta + k \cdot 2\pi$

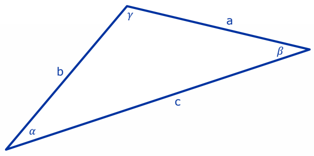
$\sin(x)^{-1} = \theta + k \cdot 2\pi \; or \; \pi - \theta + k \cdot 2\pi$

$\tan(x)^{-1} = \theta + k\pi$



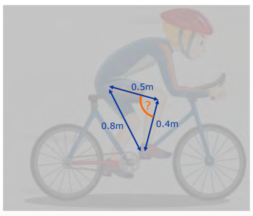
TRIGONOMETRIC FUNCTIONS
COSINUS RULE

$c^(= a^(+ b^(- 2ab\cos(\gamma)$



TRIGONOMETRIC FUNCTIONS
EXAMPLE

The distance between the hip and the foot of a cyclist is measured and is equal to 0.8m. If his upper and lower leg are respectively 0.5m and 0.4m long, what is the angle of the knee?



$c^(= a^(+ b^(- 2ab\cos(\gamma)$

$0.8^(= 0.4^(+ 0.5^(- 2 \cdot 0.5 \cdot 0.5\cos(\gamma)$

$0.23 = -0.4\cos(\gamma)$

$\cos(\gamma) = -0.575$

$\gamma = \cos(-0.575)^{-1} = 125.1°$

2 Chapter 2: Differentiation

: Chapter 2: Differentiation

2.1 First order derivatives

2.2 Second order derivatives

2.3 Partial derivatives

2.4 Additional information, slides, articles

CHAPTER 2: DIFFERENTIATION

- First order derivatives
- Second order derivatives
- Partial derivatives

VRIJE UNIVERSITEIT BRUSSEL

FIRST ORDER DERIVATIVE

EXAMPLE

A swimmer jumps from a height of 10m into a pool. The position of the swimmer is given by h, and evolves with time according to following formula:

$$h(t) = 10 - 5t^{!}$$

What is the velocity of the swimmer at each point in time?

FIRST ORDER DERIVATIVE

EXAMPLE

Option 1: Numerical differentiation

- Step 1: Calculate the vertical position of the swimmer at each point in time using the formula $h(t) = 10 - 5t^{!}$.
- Step 2: The velocity at each timepoint is the ratio of the change in position and the timesteps between two datapoints.

$$v(t) = \frac{\Delta h}{\Delta t}$$

!! This method gives the average velocity over a time interval. The velocity depends on the selected time interval.

t [s]	h [m]	v [m/s]
0	10	$\frac{9,8-10}{0,2-0} = -1$
0,2	9,8	$\frac{9,2-9,8}{0,4-0,2} = -3$
0,4	9,2	$\frac{8,2-9,2}{0,6-0,4} = -5$
0,6	8,2	$\frac{6,8-8,2}{0,8-0,6} = -7$
0,8	6,8	$\frac{5-6,8}{1-0,8} = -9$
1	5	$\frac{2,8-5}{1,2-1} = -11$
1,2	2,8	$\frac{0,2-2,8}{1,4-1,2} = -13$
1,4	0,2	/

FIRST ORDER DERIVATIVE

EXAMPLE

The estimation of the velocity is more accurate if the time between different datapoints is shorter.

t [s]	h [m]	v [m/s]
0	10	$\frac{9,8-10}{0,2-0} = -1$
0,2	9,8	$\frac{9,2-9,8}{0,4-0,2} = -3$
0,4	9,2	$\frac{8,2-9,2}{0,6-0,4} = -5$
0,6	8,2	$\frac{6,8-8,2}{0,8-0,6} = -7$
0,8	6,8	$\frac{5-6,8}{1-0,8} = -9$
1	5	$\frac{2,8-5}{1,2-1} = -11$
1,2	2,8	$\frac{0,2-2,8}{1,4-1,2} = -13$
1,4	0,2	/

t [s]	h [m]	v [m/s]
0	10	$\frac{9,95-10}{0,1-0} = -0,5$
0,1	9,95	$\frac{9,8-9,95}{0,2-0,1} = -1,5$
0,2	9,8	$\frac{9,55-9,8}{0,3-0,2} = -2,5$
0,3	9,55	$\frac{9,2-9,55}{0,4-0,2} = -3,5$
0,4	9,2	$\frac{8,75-9,2}{0,5-0,4} = -4,5$
0,5	8,75	$\frac{8,2-8,75}{0,6-0,5} = -5,5$
0,6	8,2	$\frac{7,55-8,2}{0,7-0,6} = -6,5$
0,7	7,55	$\frac{6,8-7,55}{0,8-0,7} = -7,5$
0,8	6,8	$\frac{5,95-6,8}{0,9-0,8} = -8,5$
0,9	5,95	$\frac{5-5,95}{1-0,9} = -9,5$
1	5	$\frac{3,95-5}{1,1-1} = -10,5$
1,1	3,95	$\frac{2,8-3,95}{1,2-1,1} = -11,5$
1,2	2,8	$\frac{1,55-2,8}{1,3-1,2} = -12,5$
1,3	1,55	$\frac{0,2-1,55}{1,4-1,3} = -13,5$
1,4	0,2	/

FIRST ORDER DERIVATIVE

EXAMPLE

Option 2: Analytic differentiation

The average velocity over an interval Δt of the swimmer is given by the difference quotient i.e., the change in vertical position divided by the change in time.

$$v_{/012/31} = \frac{\Delta h}{\Delta t} = \frac{h(t+\Delta t) - h(t)}{\Delta t}$$

We know that $h(t) = 10 - 5t^{!}$.

$$v_{/012/31} = \frac{10 - 5(t+\Delta t)^{!} - 10 + 5t^{!}}{\Delta t}$$

$$v_{/012/31} = \frac{-5(t^{!} + 2t\Delta t + \Delta t^{!}) + 5t^{!}}{\Delta t}$$

$$v_{/012/31} = \frac{-5(2t\Delta t + \Delta t^{!})}{\Delta t}$$

$$v_{/012/31} = -5(2t + \Delta t) = -10t - 5\Delta t$$

FIRST ORDER DERIVATIVE

EXAMPLE

Option 2: Analytic differentiation

$$v_{/012/31} = -5(2t + \Delta t) = -10t - 5\Delta t$$

If the interval $\Delta t \to 0$ (becomes very small), then the difference quotient becomes a measure of instantaneous rate of change of the function. This is called the derivative.

$$v_{4567/57/51896} = \lim_{\Delta 7 \to <} (v_{/012/31})$$

$$v_{4567/57/51896} = \lim_{\Delta 7 \to <} (-10t - 5\Delta t)$$

$$v_{4567/57/51896} = -10t$$

!! Analytic differentiation can only be used when the function is known. It can be used to calculate the instantaneous change of the function at every timepoint.

FIRST ORDER DERIVATIVE
EXAMPLE

$v_{/012/31} = -10t - 5\Delta t$

$v_{4567/57/51896} = -10t$

$\Delta t = 0,2$		
t [s]	h [m]	v [m/s]
0	10	−1
0,2	9,8	−3
0,4	9,2	−5
0,6	8,2	−7
0,8	6,8	−9
1	5	−11
1,2	2,8	−13
1,4	0,2	-15

Average velocity
Same results as slide 4

$\Delta t = 0,2$		
t [s]	h [m]	v [m/s]
0	10	−0,5
0,1	9,95	−1,5
0,2	9,8	−2,5
0,3	9,55	−3,5
0,4	9,2	−4,5
0,5	8,75	−5,5
0,6	8,2	−6,5
0,7	7,55	−7,5

Average velocity
Same results as slide 4

$\Delta t \to 0$		
t [s]	h [m]	v [m/s]
0	10	0
0,1	9,95	−1
0,2	9,8	−2
0,3	9,55	−3
0,4	9,2	−4
0,5	8,75	−5
0,6	8,2	−6
0,7	7,55	−7

Instantaneous velocity

FIRST ORDER DERIVATIVE
DEFINITIONS

The difference quotient is a measure of the average rate of change of the function over an interval.

$$\frac{\Delta f}{\Delta t} = \frac{f(t + \Delta t) - f(t)}{\Delta t}$$

The difference quotient is highly dependent on the size of the interval (Δt).

If the interval $\Delta t \to 0$ (becomes very small), then the difference quotient becomes a measure of instantaneous rate of change of the function. This is called the derivative.

$$\frac{df}{dt} = \frac{f(t + dt) - f(t)}{dt}$$

Differentiation can be done in 3 ways:

1. Analytic differentiation using the definition ➔ Used when a function f is available, slow method
2. Analytic differentiation: faster method ➔ Used when a function f is available, fast method
3. Numerical differentiation ➔ Used when a function f is not available

FIRST ORDER DERIVATIVE
ANALYTIC DIFFERENTIATION USING THE DEFINITION

$h(t) = 10 - 5 \cdot t^{!}$

Step 1: Calculate h(t) and h(t+Δt)

$h(t) = 10 - 5 \cdot t^{!}$

$h(t + \Delta t) = 10 - 5(t + \Delta t)^{!}$

$h(t + \Delta t) = 10 - 5(t^{!} + 2 \cdot t \cdot \Delta t + \Delta t^{!})$

$h(t + \Delta t) = 10 - 5(t^{!} + 2 \cdot t \cdot \Delta t + \Delta t^{!})$

$h(t + \Delta t) = 10 - 5 \cdot t^{!} - 10 \cdot t \cdot \Delta t - 5 \cdot \Delta t^{!}$

Step 2: Calculate the difference quotient.

$\frac{>?}{>7} = \frac{?\,(7@>7)A?\,(7)}{>7}$

$\frac{>?}{>7} = \frac{B<AC\cdot 7^{!}AB<\cdot 7\cdot >7AC\cdot >7^{!}AB<@C\cdot 7^{!}}{>7}$

$\frac{>?}{>7} = -10 \cdot t - 5 \cdot \Delta t$

Step 3: Calculate the derivative ($\Delta t \to 0$).

$\frac{E?}{E7} = h^{F} = -10t$

Step 4: Calculate the derivative at a certain timepoint e.g., $t = 0.8s$

$\frac{E?}{E7}(t = 0.8) = h^{F}(t = 0.8) = -8$

FIRST ORDER DERIVATIVE
ANALYTIC DIFFERENTIATION: FASTER METHOD

Rules:

1. $f(x) = a \to f^{F}(x) = \frac{GH}{GI} = 0$
2. $f(x) = x^{5} \to f^{F}(x) = \frac{GH}{GI} = nx^{5AB}$
3. $f(x) = a\,g(x) \to f^{F}(x) = \frac{GH}{GI} = a\,g'(x)$
4. $f(x) = u(x) \pm v(x) \to f^{F}(x) = u^{F}(x) \pm v^{F}(x)$
5. $f(x) = u(x) \cdot v(x) \to f^{F}(x) = u^{F}(x) \cdot v(x) + u(x) \cdot v'(x)$
6. $f(x) = \frac{9(I)}{0(I)} \to f^{F}(x) = \frac{9^{"}(I)\cdot 0(I)\,A9(I)\cdot 0^{"}(I)}{(0(I))^{!}}$
7. $f(x) = u(v(x)) \to f^{F}(x) = u^{F}(v(x)) \cdot v'(x)$

Rules for special mathematical functions:

8. $f(x) = e^{I} \to f^{F}(x) = e^{I}$
9. $f(x) = \log_{/}(x) \to f^{F}(x) = \frac{B}{I\cdot JK(/)}$
10. $f(x) = \ln(x) \to f^{F}(x) = \frac{B}{I}$
11. $f(x) = \sin(x) \to f^{F}(x) = \cos(x)$
12. $f(x) = \cos(x) \to f^{F}(x) = -\sin(x)$
13. $f(x) = \tan(x) \to f^{F}(x) = \frac{B}{NOP(I)}$
14. $f(x) = \arcsin(x) \to f^{F}(x) = \frac{B}{\sqrt{BAI^{!}}}$
15. $f(x) = \arccos(x) \to f^{F}(x) = -\frac{B}{\sqrt{BAI^{!}}}$
16. $f(x) = arctan(x) \to f^{F}(x) = \frac{B}{B@I^{!}}$

FIRST ORDER DERIVATIVE
ANALYTIC DIFFERENTIATION: FASTER METHOD

Examples:

1. $f(x) = x^{Q} \xrightarrow{R9S1\,!} f^{F}(x) = 3x^{!}$
2. $f(x) = \sqrt{x} = x^{\frac{\#}{!}} \xrightarrow{R9S1\,!} f^{F}(x) = \frac{B}{!}x^{A\frac{\#}{!}} = \frac{B}{!\sqrt{I}}$
3. $f(x) = 3x^{!} \xrightarrow[3^{"}(I)U!I]{R9S1\,Q\;T47?3(I)U\,I^{!}} f^{F}(x) = 3 \cdot 2x = 6x$
4. $f(x) = \frac{I^{!}}{!I@B} \xrightarrow[9^{"}(I)U!I\,/5G\,0F(I)U!]{R9S1\,V\;T47?9(I)UI^{!}/5G\;0(I)U!I@B} f^{F}(x) = \frac{!I^{!}@!I}{(!I@B)^{!}}$
5. $f(x) = \cos(2x) \xrightarrow[9^{"}(0(I)UA\,PXK(0(I))/5G\,0^{"}(I)U!]{R9S1\,WT47?9(0(I))UNOP(0(I))/5G\,0(I)U!I} f^{F}(x) = -\sin(v(x)) \cdot 2 = -2 \cdot \sin(2x)$

FIRST ORDER DERIVATIVE
ANALYTIC DIFFERENTIATION: FASTER METHOD

Example: The position of a point is given by $x(t) = 6te^{A!7^{!}}$. What is the velocity at t=0 and t=2?

Step 1: Calculate the derivative.

$$x(t) = 6te^{A!7^{!}}$$

$$v = \frac{dx(t)}{dt} = \frac{d(6te^{A!7^{!}})}{dt} = \frac{d(6)}{dt}te^{A!7^{!}} + 6\frac{d(t)}{dt}e^{A!7^{!}} + 6t\frac{d(e^{A!7^{!}})}{dt}$$

$$v = 6e^{A!7^{!}} + 6te^{A!7^{!}}\frac{d(-2t^{!})}{dt}$$

$$v = 6e^{A!7^{!}} + 6te^{A!7^{!}} \cdot -4t$$

$$v = (6 - 24t^{!}) \cdot e^{A!7^{!}}$$

Step 2: Calculate the derivative at t=0 and t=2

$v(t = 0) = 6$ and $v(t = 2) = -0.03$

FIRST ORDER DERIVATIVE

ANALYTIC DIFFERENTIATION: FASTER METHOD

Example: A swimmer jumps from a height of 10m into a pool. The position of the swimmer is given by $h(t) = 10 - 5t^{!}$. What is the position and the velocity at t=1 and t=0.5?

Step 1: The position can be found by inserting t=1 and t=0.5 into the equation for h.

$$h(t=1) = 10 - 5 \cdot 1^{!} = 5 \text{ and } h(t=0.5) = 10 - 5 \cdot 0.5^{!} = 8{,}75$$

Step 1: Calculate the derivative to find the velocity.

$$v(t) = \frac{dh}{dt} = \frac{d(10 - 5t^{!})}{dt} = -10t$$

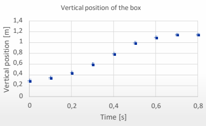
$$v(t=1) = -10 \text{ and } v(t=0.5) = -5$$

FIRST ORDER DERIVATIVE

NUMERICAL DIFFERENTIATION

Often, the relation between two variables e.g., time and position, is not given by a function but by a series of measurements. In this case, an analytic differentiation can not be performed to calculate the rate of change of a variable. Instead, a numerical differentiation is performed.

Example: A subject picks up a box. The vertical position (z) of the box is measured over time (t). What is the velocity at each timepoint?



t [s]	z [m]
0	0,3
0,1	0,35
0,2	0,45
0,3	0,6
0,4	0,8
0,5	1
0,6	1,1
0,7	1,15
0,8	1,15

FIRST ORDER DERIVATIVE

NUMERICAL DIFFERENTIATION

Often, the relation between two variables e.g., time and position, is not given by a function but by a series of measurements. In this case, an analytic differentiation can not be performed to calculate the rate of change of a variable. Instead, a numerical differentiation is performed.

Example: A subject picks up a box. The vertical position (z) of the box is measured over time (t). What is the horizontal velocity at each timepoint?

The horizontal velocity of the box can be estimated by dividing the change in horizontal position by the change in time.

$$v_4 = (z_4 - z_{4AB})/\Delta t$$

Alternative formula:

$$v_4 = (z_{4@B} - z_{4AB})/2\Delta t$$

The estimation of the velocity is more accurate if the time between the different datapoints is shorter.

i	t [s]	z [m]	v [m/s]
0	0	0,3	0
1	0,1	0,35	$\frac{0{,}35 - 0{,}3}{0{,}1} = 0{,}5$
2	0,2	0,45	$\frac{0{,}45 - 0{,}35}{0{,}1} = 1$
3	0,3	0,6	$\frac{0{,}6 - 0{,}45}{0{,}1} = 1{,}5$
4	0,4	0,8	$\frac{0{,}8 - 0{,}6}{0{,}1} = 2$
5	0,5	1	$\frac{1 - 0{,}8}{0{,}1} = 2$
6	0,6	1,1	$\frac{1{,}1 - 1}{0{,}1} = 1$
7	0,7	1,15	$\frac{1{,}15 - 1{,}1}{0{,}1} = 0{,}5$
8	0,8	1,15	$\frac{1{,}15 - 1{,}15}{0{,}1} = 0$

FIRST ORDER DERIVATIVE

PHYSICAL INTERPRETATION

When a function $f(t)$ describes the relation between two variables, then the derivative $\frac{GH}{G7} = f'(t)$ can have a physical meaning.

Examples:

- If $f(t)$ describes the relation between the position of an object and the time, then $f'(t)$ describes the relation between the velocity of an object and time.
- If $f(t)$ describes the relation between the velocity of an object and the time, then $f'(t)$ describes the relation between the acceleration of an object and time.

FIRST ORDER DERIVATIVE

MATHEMATICAL INTERPRETATION

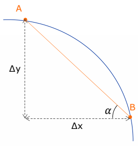
The slope of a function is defined as the ratio of the vertical change (Δy) between two points (A and B), to the horizontal (Δx) change between the same two points.

The slope corresponds to the difference quotient.

$$slope = \frac{\Delta y}{\Delta x} = \tan(\alpha)$$

If $\Delta x \to 0$, then the slope represents the derivative.

$$slope = \frac{dy}{dx} = \tan(\alpha)$$

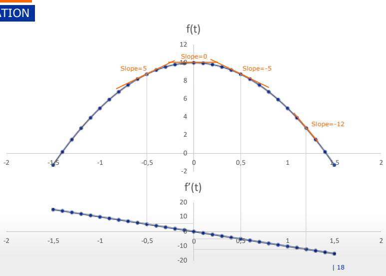


FIRST ORDER DERIVATIVE

MATHEMATICAL INTERPRETATION

So, the derivative explains how the function $f(t)$ evolves.

- If f(t) increases, then f'(t) is positive.
- If f(t) decreases, then f'(t) is negative.
- If f(t) is constant, then f'(t) is equal to 0.
- If f(t) is steep, then f'(t) is large.
- If f'(t) is 0, then f(t) has a maximum or a minimum.



SECOND ORDER DERIVATIVE
DEFINITION

First order derivative: $\frac{EH}{E7} = \frac{H(7@E7)AH(7)}{E7}$

- Can be used to calculate the velocity of an object when f(t) represent the positions of an object in time.

- Can be used to calculate the acceleration of an object when f(t) represent the velocity of an object in time.

Second order derivative: $\frac{E^!H}{E7^!} = \frac{G}{G7}\left(\frac{GH}{G7}\right)$

- Can be used to calculate the acceleration of an object when f(t) represent the position of an object in time.

$$a(t) = v^F(t) = s^{FF}(t)$$
$$a(t) = \frac{dv}{dt} = \frac{d^!s}{dt^!}$$

| 19

SECOND ORDER DERIVATIVE
EXAMPLES

$$f(x) = \ln(2x)$$
$$\frac{df(x)}{dt} = \frac{1}{2x}$$
$$\frac{d^\$f(x)}{dt^\$} = \frac{-1}{2x^\$}$$

$$f(x) = 5x^! + 1x - 3$$
$$\frac{df(x)}{dt} = 10x + 1$$
$$\frac{d^\$f(x)}{dt^\$} = 10$$

| 20

SECOND ORDER DERIVATIVE
EXAMPLE

Calculate the position, velocity and acceleration of an object at t=1, if the position of the object is given by: $f(t) = -7t^C + 4t^! - 23$

Step 1: Calculate the position

$$f(t=1) = -7 \cdot 1^C + 4 \cdot 1^! - 23 = -26$$

Step 2: Calculate the velocity

$$v(t) = \frac{df(t)}{dt} = -35t^Y + 8t$$
$$v(t=1) = -35 \cdot 1^Y + 8 \cdot 1 = -27$$

Step 3: Calculate the acceleration

$$a(t) = \frac{d^!f(t)}{dt^!} = \frac{dv(t)}{dt} = -140t^Q + 8$$
$$a(t=1) = -140 \cdot 1^Q + 8 = -132$$

| 21

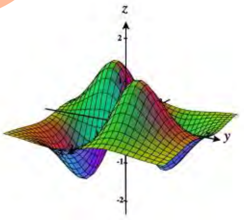
PARTIAL DERIVATIVE
DEFINITION

Sometimes a function depends on multiple variables. Example: During running, the angle of the knee depends on the time, the inclination of the floor, …

$$angle = f(time, inclination, \ldots) = f(t, i, \ldots)$$

If we would be interested in the rate of change of the function (e.g., angle of the knee) with respect to one of the variables (e.g., the time), then we would take the partial derivative. Here we consider all other variables (e.g., inclination of the floor) constant, and we take the derivative with respect to the variable of interest.

$$\frac{\partial f(t, i, \ldots)}{\partial t}$$



| 22

PARTIAL DERIVATIVE
EXAMPLES

Example:

$$f(x,y) = x^! + xy + 3y^Q$$
$$\frac{\partial f(x,y)}{\partial x} = 2x + y$$
$$\frac{\partial f(x,y)}{\partial y} = x + 9y^!$$

$$f(x,y,z) = xy^Q + 3xz - 4yz^Y$$
$$\frac{\partial f(x,y,z)}{\partial x} = y^Q + 3z$$
$$\frac{\partial f(x,y,z)}{\partial y} = 3xy^! - 4z^Y$$
$$\frac{\partial f(x,y,z)}{\partial z} = 3x - 16yz^Q$$

| 23

3 Chapter 3: Integration

: Chapter 3: Integration

3.1 Primitive functions

3.2 Applications in kinematics

3.3 Additional information, slides, articles

CHAPTER 3: INTEGRATION

► Primitive function

► Applications in kinematics

VRIJE UNIVERSITEIT BRUSSEL

PRIMITIVE FUNCTION

DEFINITION

Remember from chapter 2: The function describing the velocity of an object is the **derivative** of the function descibing the position of the object. The derivative can be found with **differentiation**.

- Example: if $x(t) = 2t^{!}$, then $v(t) = \frac{"\#(\$)}{"\$} = 4t$.

The function describing the position of an object is called the **primitive function** of the velocity function. The primitive function can be obtained by **integration**. Multiple primitive functions exist.

- Example: if $v(t) = 4t$, then $x(t) = \int v(t)dt = 2t^{!} + c =$ - $x(t) = 2t^{!}$, $x(t) = 2t^{!} + 4$, $x(t) = 2t^{!} - 3$.

Note that the derivative of the primitive function of the velocity function, is the velocity function itself.

- Example: if $v(t) = 4t$, then $x(t) = \int v(t)dt = 2t^{!} + c =$ - $x(t) = 2t^{!}$, $x(t) = 2t^{!} + 4$, $x(t) = 2t^{!} - 3$ and $\frac{"\#(\$)}{"\$} = 4t$.

PRIMITIVE FUNCTION

DEFINITION

If F(t) is the primitive function of f(t), then $F^{\%}(t) = \frac{"\&(\$)}{"\$} = f(t)$ and $F(t) = \int f(t)dt$

Rules:

1. $f(x) = x^{'} \rightarrow F(x) = \int f(x)dx = \frac{(}{')(}x^{')(} + k$
2. $f(x) = c \rightarrow F(x) = \int f(x)dx = cx + k$
3. $f(x) = \sin(x) \rightarrow F(x) = \int f(x)dx = -\cos(x) + k$
4. $f(x) = \cos(x) \rightarrow F(x) = \int f(x)dx = \sin(x) + k$
5. $f(x) = \frac{(}{*+,!(\#)} \rightarrow F(x) = \int f(x)dx = \tan(x) + k$
6. $f(x) = \frac{(}{\#} \rightarrow F(x) = \int f(x)dx = \ln|x| + k$
7. $f(x) = e^{\#} \rightarrow F(x) == \int f(x)dxe^{\#} + k$
8. $\int (f(x) \pm g(x))dx = \int f(x)dx \pm \int g(x)dx$
9. $\int af(x)dx = a\int f(x)dx$

PRIMITIVE FUNCTION

EXAMPLES

1. $\int (2x)dx = 2\int xdx = \frac{!}{!}x^{!} = x^{!}$
2. $\int \left(\frac{-}{\#} + x\right) dx = \int \left(\frac{-}{\#}\right) dx + \int xdx = 3\int \left(\frac{(}{\#}\right) dx + \frac{\#^{!}}{!} + k_{(} = 3\ln(x) + k_{!} + \frac{\#^{!}}{!} + k_{(} = 3\ln(x) + \frac{\#^{!}}{!} + k'$
3. $\int 6x^{-} dx = 6\int x^{-} dx = 6 \cdot \left(\frac{(}{.}x^{\cdot} + k\right) = \frac{-}{!}x^{\cdot} + 6k = \frac{-}{!}x^{\cdot} + k^{\%}$
4. $\int \frac{!}{\$} dt = 2\int \frac{(}{\$} dt = 2(\ln(t) + k) = 2\ln(t) + k^{\%}$
5. $\int \frac{(}{/^{!}} dy = \int (y^{0!})dy = \frac{(}{0!)(}y^{0!)(} + k = -y^{0(} + k = -\frac{(}{/} + k$

APPLICATIONS IN KINEMATICS

EQUATIONS OF MOTION

Kinematics describes motion/movement in function of positions x(t), velocities v(t) and accelerations a(t).

We already know:

$$v(t) = \frac{dx(t)}{dt}$$

$$a(t) = \frac{dv(t)}{dt} = \frac{d^{!}x(t)}{dt^{!}}$$

and

$$x(t) = \int v(t)dt$$

$$v(t) = \int a(t)dt$$

These equations can be used to describe motion.

APPLICATIONS IN KINEMATICS

EQUATIONS OF MOTION

Example: An object has a constant acceleration $a(t) = a_1$ and moves in a straight path. Determine the position x(t) of the object in function of the time.

$$a(t) = a_1$$

$$v(t) = \int a(t)dt = \int a_1 dt = a_1 t + k$$

We know that the velocity at $t = 0$ equal is to v_1.

$$v(t = 0) = v_1 = a_1 \cdot 0 + k \rightarrow k = v_1$$

$$v(t) = a_1 t + v_1$$

$$x(t) = \int v_{\$} dt = \int (a_1 t + v_1)dt = a_1 t^{!} + v_1 t + k$$

We know that the position at $t = 0$ equal is to x_1.

$$x(t = 0) = x_1 = a_1 \cdot 0^{!} + v_1 \cdot 0 + k \rightarrow k = x_1$$

$$x(t) = a_1 t^{!} + v_1 t + x_1$$

APPLICATIONS IN KINEMATICS
EQUATIONS OF MOTION

Example: An object has a acceleration given by $a(t) = \cos(2t)$ and moves in a straight path.The velocity at $t = 0$ is equal to zero, and the position at $t = 0$ is also zero. Determine the position x(t) of the object in function of the time.

$$a(t) = \cos(2t)$$

$$v(t) = \int a(t)dt = \int \cos(2t)dt = \frac{1}{2}\sin(2t) + k$$

We know that the velocity at $t = 0$ equal is to 0.

$$v(t = 0) = 0 = \frac{1}{2}\sin(2 \cdot 0) + k \rightarrow k = 0$$

$$v(t) = \frac{1}{2}\sin(2t)$$

$$x(t) = \int v(t)dt = \int \left(\frac{1}{2}\sin(2t)\right)dt = -\frac{1}{4}\cos(2t) + k$$

We know that the position at $t = 0$ equal is to 0.

$$x(t = 0) = 0 = -\frac{1}{4}\cos(2 \cdot 0) + k \rightarrow k = \frac{1}{4}$$

$$x(t) = -\frac{1}{4}\cos(2t) + 1/4$$

| 7

APPLICATIONS IN KINEMATICS
EQUATIONS OF MOTION

Exercise: The position of an object, moving on a straight path, is given by: $x(t) = 2t^{\text{-}} - 6t^{!} + 4$

A. Calculate when the velocity is equal to 18 m/s.

B. Calculate when the velocity is equal to -6 m/s.

C. Calculate at which moment the object slows down after an acceleration. Does, at this moment the direction of movement changes?

| 8

APPLICATIONS IN KINEMATICS
EQUATIONS OF MOTION

Exercise: The position of an object, moving on a straight path, is given by: $x(t) = 2t^{\text{-}} - 6t^{!} + 4$

A. Calculate when the velocity is equal to 18 m/s.

$$x(t) = 2t^{\text{-}} - 6t^{!} + 4$$
$$v(t) = 6t^{!} - 12t = 18$$
$$0 = 6t^{!} - 12t - 18$$
$$0 = t^{!} - 2t - 3$$
$$t = -1$$

B. Calculate when the velocity is equal to -6 m/s.

$$v(t) = 6t^{!} - 12t = -6$$
$$0 = 6t^{!} - 12t + 6$$
$$0 = t^{!} - 2t + 1$$
$$t = 1$$

| 9

APPLICATIONS IN KINEMATICS
EQUATIONS OF MOTION

Exercise: The position of an object, moving on a straight path, is given by: $x(t) = 2t^{\text{-}} - 6t^{!} + 4$

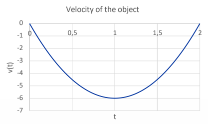
C. Calculate at which moment the object accelerates after slowing down. Does, at this moment the direction of movement changes?

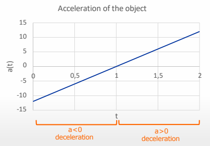
$$x(t) = 2t^{\text{-}} - 6t^{!} + 4$$
$$v(t) = 6t^{!} - 12t$$
$$a(t) = 12t - 12$$
$$a(t) = 12t - 12 = 0$$

At $t = 1$, the acceleration becomes 0.

$$v(t = 1) = 6 \cdot 1^{!} - 12 \cdot 1 = -6$$

The direction of movement does not change since the velocity is negative before and after t=1s.





| 10

APPLICATIONS IN KINEMATICS
EQUATIONS OF MOTION

Exercise: The acceleration of an object, moving on a straight path, is given by: $a(t) = 3e^{0!\$}$. Calculate $x(t = 1)$ and $x(t = 10)$ if you know that $x(t = 0) = 0$ and $v(t = 0) = 0$.

$$a(t) = 3e^{0!\$}$$

$$v(t) = \int a(t)dt = -\frac{3}{2}e^{0!\$} + k_1$$

$$v(t = 0) = 0 = -\frac{3}{2}e^{0!\cdot 1} + k_1 \rightarrow k_1 = \frac{3}{2}$$

$$v(t) = -\frac{3}{2}e^{0!\$} + \frac{3}{2}$$

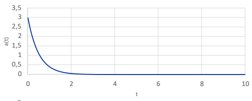
$$x(t) = \int x(t)dt = \frac{3}{4}e^{0!\$} + \frac{3}{2}t + k_{(}$$

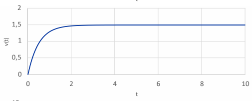
$$x(t = 0) = \frac{3}{4}e^{0!\cdot 1} + \frac{3}{2} \cdot 0 + k_{(} = \frac{3}{4} + k_{(} \rightarrow k_{(} = -\frac{3}{2}$$

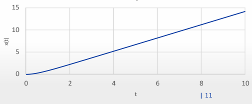
$$x(t) = \frac{3}{4}e^{0!\$} + \frac{3}{2}t - \frac{3}{4}$$

$$x(t = 1) = 0.85$$

$$x(t = 10) = 14{,}25$$







| 11

APPLICATIONS IN KINEMATICS
EQUATIONS OF MOTION

Exercise: The velocity of an object, moving on a straight path, is given by: $v(t) = \frac{!}{\$} + 6$. Calculate $x(t = 2)$ if you know that $x(t = 1) = 3$.

$$v(t) = \frac{2}{t} + 6$$

$$x(t) = \int \left(\frac{2}{t} + 6\right)dt = 2\ln(t) + 6t + k$$

$$x(t = 1) = 3 = 2\ln(1) + 6 + k = 6 + k \rightarrow k = 3$$

$$x(t) = 2\ln(t) + 6t + 3$$

$$x(t = 2) = 2\ln(2) + 6 \cdot 2 + 3 = 2\ln(2) + 15$$

| 12

4 Chapter 4: Coordinates and coordinate systems

: Chapter 4: Coordinates and coordinate systems

4.1 2D coordinate systems

4.1.1 Cartesian coordinates

4.1.2 Pole coordinates

4.2 3D coordinate systems

4.2.1 Cartesian coordinates

4.2.2 Homogeneous coordinates

4.3 Additional information, slides, articles

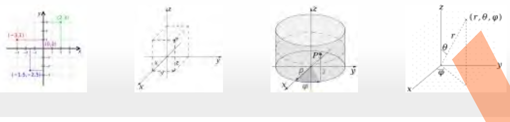
CHAPTER 4: COORDINATES AND COORDINATE SYSTEMS

COORDINATES AND COORDINATE SYSTEMS
DEFINITION

In mathematics, the positions of points and objects are described with respect to a **reference frame**. This reference frame consists out of axes e.g. the x- and y-axis, and an origin located where the axes meet.

Angles and distances can be used to describe where a point or object is located in the reference frame. The angles and distances are called **coordinates**.

Describing the position of points can be done with different **coordinate systems** e.g., cartesian coordinate systems in two or three dimensions, cylindrical or spherical coordinate systems, …

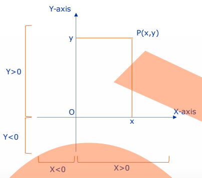


2D COORDINATE SYSTEMS
CARTESIAN COORDINATES

The 2D Cartesian coordinate system consists of two axes (x and y) and the origin O, located where the axes intersect. The x-axis is mostly drawn horizontally to the right, the y-axis is then drawn vertically upwards.

A position of a point in a 2D Cartesian coordinate system is given by two coordinates: x and y.

- X represents the distance between the origin O and the projection of the point P on the x-axis. The x-coordinate is positive when the point P is located to the right of the y-axis, else the coordinate is negative.
- Y is the distance between the origin O and the projection of the point P on the y-axis. The y-coordinate is positive when the point P is located above the x-axis, else the coordinate is negative.

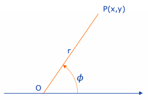


2D COORDINATE SYSTEMS
POLE COORDINATES

The pole coordinate systems consists of one axis and an origin O.

The position of a point P is given by two coordinates: the angle ϕ and the distance r.

- r is the distance between the point P and the origin O. r is always positive.
- ϕ is the angle between the axis and the line OP (connecting O and P). Draw an arrow from the positive side of the axis towards line OP. If the arrow goes counterclockwise, then $\phi > 0$, else $\phi < 0$.

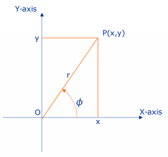


2D COORDINATE SYSTEMS
CARTESIAN VS POLE COORDINATE

There is a relation between Cartesian and pole coordinates:

$$x = r \cdot \cos(\phi)$$
$$y = r \cdot \sin(\phi)$$

$$r = \sqrt{x^2 + y^2}$$
$$\phi = \arctan\left(\frac{y}{x}\right) = tan^{-1}\left(\frac{y}{x}\right)$$

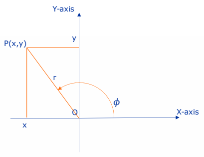


2D COORDINATE SYSTEMS
CARTESIAN VS POLE COORDINATE

Example: A point P has Cartesian coordinates x=-3 and y=4. What are the pole coordinates?

$$r = \sqrt{(-3)^2 + 4^2} = 5$$
$$\phi = \arctan\left(\frac{4}{-3}\right) = -53.15° + k \cdot 180°$$
$$\phi = -53.15° \; or \; 126.87°$$

From the drawing we know that the angle is 126.87°.



2D COORDINATE SYSTEMS
CARTESIAN VS POLE COORDINATE

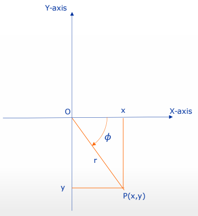
Example: A point P has Cartesian coordinates x=3 and y=-4. What are the pole coordinates?

$$r = \sqrt{(-4)^{!}+3^{!}} = 5$$

$$\phi = \arctan\left(\frac{-4}{3}\right) = -53.15° + k \cdot 180°$$

$$\phi = -53.15°\ or\ 126.87°$$

From the drawing we know that the angle is $-53.15°$.



2D COORDINATE SYSTEMS
CARTESIAN VS POLE COORDINATE

Exercise: Calculate the pole coordinates from following points.

1. P(x,y)=(2,-2)
2. Q(x,y)=(6,8)
3. R(x,y)=(0,5)
4. S(x,y)=(5,0)

Exercise: Calculate the cartesian coordinates from following points.

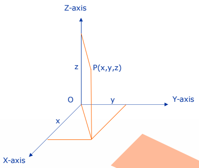
1. A(r,ϕ)=(3,-60°)
2. B(r,ϕ)=(3,60°)
3. C(r,ϕ)=(0,60°)
4. D(r,ϕ)=(3,0°)

3D COORDINATE SYSTEMS
CARTESIAN COORDINATES

The 3D Cartesian coordinate system consists of three axes (x, y and z) and the origin O located where the axes intersect.

A position of a point in a 3D Cartesian coordinate system is given by three coordinates: x, y and z.

- X represents the distance between the origin O and the projection of the point P on the x-axis.
- Y is the distance between the origin O and the projection of the point P on the y-axis.
- Z is the distance between the origin O and the projection of the point P on the z-axis.



3D COORDINATE SYSTEMS
HOMOGENEOUS COORDINATES

Cartesian coordinates (x,y,z) ➔ Homogeneous coordinates (p,q,r,s)

$$x = \frac{p}{s}$$

$$y = \frac{q}{s}$$

$$z = \frac{r}{s}$$

Note, a representation in homogenous coordinates is not unique. A same point can be described with different coordinates.

If s=1 (normalized homogenous coordinates), then the point with cartesian coordinates (x,y,z) has homogenous coordinates (x,y,z,1).

Homogenous coordinates can also be used in 2D situations. Then cartesian coordinates (x,y) correspond to homogenous coordinates (a,b,c).

3D COORDINATE SYSTEMS
EXERCISES

5 Introduction into programming

: An Introduction into Programming

5.1 Python language

5.1.1 Python as a smart calculator

5.1.2 Lists in Python

5.1.3 Strings in Python

5.1.4 NumPy – NUMerical PYthon

5.1.5 Plotting graphs in Python

5.2 Additional information, slides, articles

An Introduction into Programming
The very basics of Python

Bart Jansen

Who Am I?

- Prof. Dr. Bart Jansen, bjansen@etrovub.be
- Prof at Department of Electronics and Informatics (ETRO) of the Vrije Universiteit Brussel (VUB)
- PhD in Computer Science, Artificial Intelligence
- Coordinating research on signal processing and AI in medical applications.
- A lot of interest in Rehabilitation Engineering

Objectives of the coming weeks

- To learn the basics of the Python language.
- To learn to express simple ideas in Python.
- To refresh some rusty mathematical concepts.
- Next semester, we will need all three ...

Expected knowledge

Expected efforts

- You program as much as ever possible.
- Programming is easy once you can do it ;-)
- There is only one way and it is the hard way.
- Get a python environment today!
  (Pycharm, Anaconda, Canopy, Wing, Eclipse ...)

Course Material: Optional

Part 1: Python as a smart calculator

What is programming?

Programming is writing a computer program by means of a series of instructions that the computer can perform.

Computing with numbers

Example 1

```
x=5
y=6
z=x*y
print(x,y,z)
```

5 6 30

x, y and z are variables.
x is assigned to be the value of 5.
y is assigned to be the value of 6.
z is assigned to be value of the multiplication of the value of x and the value of y.

[redacted]

We assume this is Python 3.X. In 2.X, last line should be print x,y,z

Variables

- Names for variables can contain small letters as well as capitals. Be coherent, it improves readability.
- Names of variables can contain numbers, but cannot start with numbers. 'x2' is ok but '2x' is not valid.
- Names of variables cannot contain any space or special characters (at least not for now).
- Think about naming your variables.

Computing with numbers

7 arithmetic operators

+	Sum
−	Subtraction
∗	Multiplication
/	Division
%	Modulus
∗∗	Exponent
//	Integer division

More operations

```
import math

x = math.cos(math.pi)
y = math.cos(math.pi / 2)

print(x,y)
```

-1.0 6.12323399574e-17

- Additional math operations are in the math module
- Use the import keyword to load a module
- All operations from a module must have the correct prefix

Or alternatively ...

```python
from math import *

x = cos(pi)
y = cos(pi / 2)

print(x,y)
```

-1.0 6.12323399574e-17

Computing something real

The length of the third side of a triangle: $a^2 + b^2 - 2ab\cos(\theta) = c^2$

```python
import math

a = 3.0
b = 4.0
theta = math.pi / 3
c = math.sqrt(a*a+b*b-2*a*b*math.cos(theta))
print(c)
```

The fact this works says something about the priority of the arithmetic operators !
What, why?

Computing something real

The length of the third side of a triangle, once more

```python
import math

a = 2.0
b = 3.0
theta = math.pi / 5
c = math.sqrt(a*a+b*b-2*a*b*math.cos(theta))
print(c)
```

But are we going to write the same program whenever we compute this for different triangles?

Functions

```python
import math

def angle(a,b,theta):
    c = math.sqrt(a*a+b*b-2*a*b*math.cos(theta))
    return c

result1 = angle(3,4,math.pi/2)
result2 = angle(2,3,math.pi/5)

print(result1, result2)
```

- The concept of a function is similar to the mathematical concept.
- The indentation matters. The tab defines what is part of the function and what is not.

Functions

```python
def functionName(list of formal parameters):
    body of the function
    returnvalue = more function body
    return returnvalue

someValue = functionName(list of actual parameters)
```

When calling the function, the actual parameters are bound to the formal parameters. The return value of the function can also be assigned to a variable.

Question

Can we have functions without return values? Why would that make any sense?

Function Example

```
def sum(x,y):
    return x+y

def prod(x,y):
    return x*y

print(sum(prod(1,2),prod(3,4)))
```


Function Example

```
def celciusToFahrenheit(temp):
    return 1.8 * temp + 32

foo = celciusToFahrenheit(0)
print(foo)
```


Function calls can be nested

```
def sum(x,y):
    return x+y

def prod(x,y):
    return x*y

def celciusToFahrenheit(temp):
    return sum(prod(1.8, temp),32)

foo = celciusToFahrenheit(0)
print(foo)
```


Function definitions can be nested

```
def celciusToFahrenheit(temp):
    def sum(x,y):
        return x+y

    def prod(x,y):
        return x*y

    return sum(prod(1.8, temp),32)

foo = celciusToFahrenheit(0)
print(foo)
```

Be very careful with the tabs !!!

An if-test

```
def compare(a,b):
    if(a>=b):
        print("first is biggest")
    else:
        print("second is biggest")

print compare(18,5)
```


Loops

```
for x in range(0,10):
    print(x)

for x in range(10,0,-2):
    print(x)
```

OMG

- We have cut some corners.
- We have been a bit quick.
- We clearly forgot to give examples.
- BUT: With variables, functions, if and for, we can actually program!
- These are core concepts, so let us practise a lot!

Part 2: Lists in python

Lists

```
pyList = [4,12,2,34,17]

first = pyList[0]
second = pyList[1]

print first, second
pyList[0] = 11
print(pyList)
```

A list is simply a sequence of data.

Adding elements to Lists

```
pyList = [4,12,2,34,17]

pyList.append(50)
pyList.append(18)
pyList.append(64)

print(pyList)
```

Elements are added at the end.

Appending two Lists

```
pyListA = [34,12]
pyListB = [4,6,31,9]

pyListA = pyListA + pyListB

print(pyListA)
```


Removing an element from a list

```
pyList = [1,2,3,4]
print len(pyList)

del pyList[1]

print pyList
```

Getting the last element in the list

```
pyList = [1,2,3,4]

list[len(pyList) - 1] = 44

print pyList

pyList[-1] = 444

print(pyList)
```

Quick and easy but mind the readability of the code

List Slicing to access and replace ranges

```
pyList = [1,2,3,4]

z = pyList[1:2]

pyList[1:2] = [55,66]

print(z)
print()
print(pyList)
```

Slices make a copy!
$[a : b]$ means from a to b, without b
$[0 : b]$ can be written as $[: b]$
$[a : end]$ can be written as $[a :]$
$[0 : end]$ is hence $[:]$

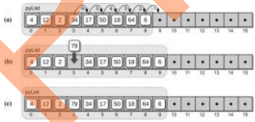
List slicing to remove ranges

```
pyList = [1,2,3,4]

z = pyList[1:2]

pyList[1:2] = []

print(z)
print()
print(pyList)
```

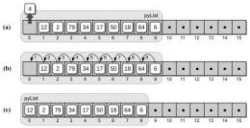

Inserting and removing a given element

```
pyList = [4,12,2,34,17,50,18,64,6]
pyList.insert(2,79)
print(pyList)
pyList.remove(4)
print(pyList)
```


Inserting 79



Removing 4



More on slices

$seq = L[start : stop : step]$ from start, to stop, every step element
$seq = L[:: 2]$ get every other item, starting with the first
$seq = L[1 :: 2]$ get every other item, starting with the second

List operations according to docs.python.org

- $list.append(x)$. Add an item to the end of the list; equivalent to $a[len(a) :] = [x]$.
- $list.extend(L)$. Extend the list by appending all the items in the given list; equivalent to $a[len(a) :] = L$.
- $list.insert(i, x)$. Insert an item at a given position. The first argument is the index of the element before which to insert, so $a.insert(0, x)$ inserts at the front of the list, and $a.insert(len(a), x)$ is equivalent to $a.append(x)$.
- $list.remove(x)$. Remove the first item from the list whose value is x. It is an error if there is no such item.

List operations according to docs.python.org

- $list.pop([i])$. Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)
- $list.index(x)$ Return the index in the list of the first item whose value is x. It is an error if there is no such item.
- $list.count(x)$ Return the number of times x appears in the list.
- $list.sort()$ Sort the items of the list, in place.
- $list.reverse()$ Reverse the elements of the list, in place.

Attention: Multiple variables can point to the same list

```
pyList = [11,12,13,14]
pyList2 = list

pyList[3] = 123
print(pyList)
print(pyList2)

del list
print(pyList2)
```

Changes to one variable can have effects elsewhere.
del deletes the pointer, not the content.

Looping over the elements

```
for x in [1,2,3,4]:
    print(x)
```

Same for loop pattern as earlier, the range function was actually simply creating a list. (not 100% correct)

TODO

- Make a function that counts the number of elements in the list using a loop.
- Do this with recursion.
- Make a function that counts how many times a given element occurs in a list.
- Do this with recursion.

Strings in Python

Strings

```
x = "hello"
y = "world"

z = x + " " + y

print(z)
```


Strings

```
x = "hello"

print(x[0])

x = x + '!'
print(x)
```

Strings behave like lists of characters, but they are not ...

Strings

```
x = "hello"

x[1] = 'y'
```

TypeError: 'str' object does not support item assignment

Strings

```
x = str(1)
y = str(1.0)
z = str(1==1)
t = str([1,2])
print(x,y,z,t)
```

1 1.0 True [1, 2]

str generates a string from any other type

Strings

```
x = int("1")
y = float("1.0")

print(x,y)
```

Something on algorithms and time complexity

Functions

```
def smallest(alist):s
  mallest = 1000000 fo
  r value in alist:
    if (value < smallest):
      smallest = value
  return smallest

print(smallest([1,2,-3,9]))
```

Finding the smallest element in a list.

Functions

```
def smallest2(alist):
  smallest = 1000000
  smallestPos = -1
  i=0
  for value in alist:
    if (value < smallest):
      smallest = value
      smallestPos = i
    i = i+1
  return smallest, smallestPos

print(smallest2([1,2,-3,9]))
```

Functions allow for multiple return values. Return values are packed into tuples.

Tuples

- Just like strings, tuples are immutable.
- x = 1,2,3 creates a tuple of three elements.
- x = 1, creates a tuple of one element.
- x = () creates a tuple of zero elements.

Functions

```
def smallest3(alist):
  smallest = 1000000
  smallestPos = -1
  for pos,value in enumerate(alist):
    if (value < smallest):
      smallest = value
      smallestPos = pos
  return smallest, smallestPos


print(smallest3([1,2,-3,9]))
```

Finding the smallest element in a list. In order to get rid of the explicit index variable, we use an enumeration.

Finding the two smallest numbers in a list

1. Find the smallest element and its position, remove it. Find the smallest element and its position. Insert the first back.
2. Sort the list, find the two smallest ones. Find their positions in the unsorted list.
3. Traverse the list once and keep indices of the two smallest elements.

Approach 1

```python
def find_two_smallest(alist):
    smallest = min(alist)
    min1 = alist.index(smallest)

    alist.remove(smallest)

    next_smallest = min(alist)
    min2 = alist.index(next_smallest)

    alist.insert(min1, smallest)

    if min1 <= min2:
        min2 += 1

    return (min1, min2)
```


Approach 2

```python
def find_two_smallest(alist):
    temp_list = alist[:]
    temp_list.sort()

    smallest = temp_list[0]
    next_smallest = temp_list[1]

    min1 = alist.index(smallest)
    min2 = alist.index(next_smallest)

    return (min1, min2)
```


Approach 3

```python
def find_two_smallest(alist):
    if alist[0] < alist[1]:
        min1, min2 = 0, 1
    else:
        min1, min2 = 1, 0

    for i in range(2, len(alist)):
        if alist[i] < alist[min1]:
            min2 = min1
            min1 = i
        elif alist[i] < alist[min2]:
            min2 = i

    return (min1, min2)
```


Approach 3b

```python
def find_two_smallest(alist):
    if alist[0] < alist[1]:
        min1, min2 = 0, 1
    else:
        min1, min2 = 1, 0

    for (i, v) in enumerate(alist):
        if v < alist[min1]:
            min2 = min1
            min1 = i
        elif v < alist[min2]:
            min2 = i

    return (min1, min2)
```


Comparison for 1400 elements

1

Algorithm	Running Time (ms)
Find, remove, find	1.117
Sort, index	2.128
Walk through list	1.472

But ... Timecomplexities

NumPy - NUMerical PYthon

Why NumPy?

- Optimized C code - NumPy's core functions are implemented in C language.
- Vectorization - operations are vectorized, reducing the need for explicit loops, indexation, ...
- Efficient data structures - uses arrays, which are more memory efficient and faster than Python lists.
- Optimized memory management.

Performance Comparison: Lists vs. NumPy Arrays

Element-Wise Multiplication Performance - Lists

```python
import time
import sys

a_list = list(range(1, 1000000))

start_time = time.time()
multiplication_list_result = [x * 2 for x in a_list]
end_time = time.time() - start_time

# Measuring memory usage for the list
list_memory = sys.getsizeof(multiplication_list_result)

print("Time taken for list operation: {:.5f} seconds".format(end_time))
print("Memory usage for list: {:.2f} KB".format(list_memory / 1024))
```

Time taken for list operation: 0.04103 seconds.
Memory usage for list: 8250.71 KB.

Performance Comparison: Lists vs. NumPy Arrays

Element-Wise Multiplication Performance - NumPy

```python
import numpy as np
import time, sys

an_array = np.array(range(1, 1000000))

start_time = time.time()
multiplication_numpy_result = an_array * 2
end_time = time.time() - start_time

# Measuring memory usage for the NumPy array
numpy_memory = sys.getsizeof(multiplication_numpy_result)

print("Time taken for NumPy operation: {:.5f} seconds".format(end_time))
print("Memory usage for NumPy: {:.2f} KB".format(numpy_memory / 1024))
```

Time taken for NumPy operation: 0.00130 seconds.
Memory usage for NumPy: 7812.59 KB

When dealing with larger datasets, the efficiency gap between lists and NumPy arrays becomes very clear...!!

Plotting graphs in Python

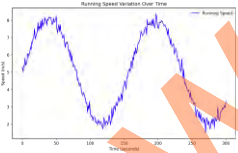
Plotting is important:

- Allows you to visually represent complex data (patient progress, motion, pain levels, etc.).
- Identify trends and patterns in patient data over time (i.e.: crucial to track changes in treatment plans).
- Facilities information exchange, allows to communicate findings with patients, colleagues.

Walking/Running speed over time - Line plot

```python
import matplotlib.pyplot as plt
import numpy as np

# Define time in seconds for 5 minutes (300 seconds)
time_seconds = np.arange(0, 301, 1)

# Simulatation of speed data
speed_data = 5 + 3 * np.sin(0.04 * time_seconds) + np.random.normal(loc=0.0, scale=0.2,
             size=len(time_seconds))

# Create a line plot
plt.figure(figsize=(10, 6))
plt.plot(time_seconds, speed_data, linestyle='-', color='b', label='Running Speed')

# Add labels and a title
plt.xlabel('Time (seconds)')
plt.ylabel('Speed (m/s)')
plt.title('Running Speed Variation Over Time')

# Add a legend
plt.legend()

# Set a high DPI value (e.g., 300) for better image quality
plt.savefig('running-speed-plot.png', dpi=400, bbox_inches='tight')

# Display the plot
plt.grid(True)
plt.show()
```


Walking/Running speed over time - Line plot



Remember...

- You program as much as ever possible.
- Programming is easy once you can do it ;-)
- There is only one way and it is the hard way.

5.3 Some random topics in mathematics

: Some random topics from mathematics

5.4 Additional information, slides, articles

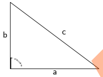
Some random topics from mathematics

Bart Jansen

September 2023

Computing with numbers

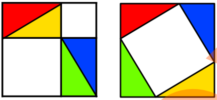
Pythagoras: $a^2 + b^2 = c^2$



Given two sides, compute the third side

Computing with numbers

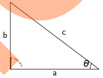
Pythagoras: $a^2 + b^2 = c^2$



https://www.math.union.edu/_dpvc/math/Pythagorus/welcome.html

Computing with numbers

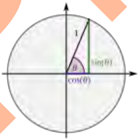
sin, cos and tan



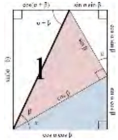
$$\sin(\theta) = \frac{b}{c}$$
$$\cos(\theta) = \frac{a}{c}$$
$$\tan(\theta) = \frac{b}{a}$$

The unit circle

so $\cos^2(\theta) + \sin^2(\theta) = 1$



The sum of angles formulas



$$\sin(\alpha + \beta) = \cos(\alpha)\sin(\beta) + sin(\alpha)\cos(\beta)$$
$$\cos(\alpha + \beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$$

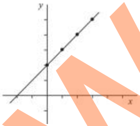
The sum of angles formulas

$$\begin{aligned}\tan(\alpha+\beta) &= \frac{\sin(\alpha+\beta)}{\cos(\alpha+\beta)} \\ &= \frac{\cos(\alpha)\sin(\beta)+\sin(\alpha)\cos(\beta)}{\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta)} \\ &= \frac{\frac{\cos(\alpha)\sin(\beta)+\sin(\alpha)\cos(\beta)}{\cos(\alpha)\cos(\beta)}}{\frac{\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta)}{\cos(\alpha)\cos(\beta)}} \\ &= \frac{\frac{\cos(\alpha)\sin(\beta)}{\cos(\alpha)\cos(\beta)}+\frac{\sin(\alpha)\cos(\beta)}{\cos(\alpha)\cos(\beta)}}{\frac{\cos(\alpha)\cos(\beta)}{\cos(\alpha)\cos(\beta)}-\frac{\sin(\alpha)\sin(\beta)}{\cos(\alpha)\cos(\beta)}} \\ &= \frac{\tan(\beta)+\tan(\alpha)}{1-\tan(\alpha)\tan(\beta)}\end{aligned}$$

The equation of a line

$y = x + 2$

x	y
0	2
1	3
2	4
3	5



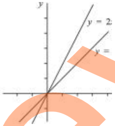
https://www.mathcentre.ac.uk/resources/uploaded/mc-ty-strtlines-2009-1.pdf

Another line

$y = 2x$

$y = 2x$:

x	y
0	0
1	2
2	4



https://www.mathcentre.ac.uk/resources/uploaded/mc-ty-strtlines-2009-1.pdf

The equation of a line

$y = mx + b$

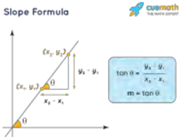
The slope (or gradient) of a line is $m = \frac{y_2 - y_1}{x_2 - x_1}$.
The equation of a straight line with gradient m and intercept c on the y-axis is $y = mx + c$.

The equation of a line with a given slope through a given point

$y - y_1 = m(x - x_1)$

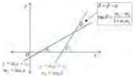
Given that a line goes through the point (x_1, y_1) and has a slope m, what is the equation of the line?
First of all, we know the generic equation $y = mx + c$, but the given point must be on the line, so $y_1 = mx_1 + c$, so $c = y_1 - mx_1$. Combining the two: $y = mx + c$ and $c = y_1 - mx_1$, we get: $y = mx + y_1 - mx_1$ or $y = m(x - x_1) + y_1$ or $y - y_1 = m(x - x_1)$.
Q: what is then the equation of a line through two given points?

Linking the slope to the tangent



https://www.cuemath.com/slope-formula/

The angle between two lines



Let the first line be given by $y = m_1x + c_1$ and the second line by $y = m_2x + c_2$, then $\alpha = \tan^{-1}(m_1)$ and $\beta = \tan^{-1}(m_2)$.

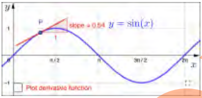
$$
\begin{aligned}
\tan(\theta) &= \tan(\beta - \alpha) \\
&= \frac{\tan(\beta) - \tan(\alpha)}{1 + \tan(\alpha)\tan(\beta)} \\
&= \frac{m_2 - m_1}{1 + m_1m_2}
\end{aligned}
$$

When two lines are perpendicular, $m_1m_2 = -1$

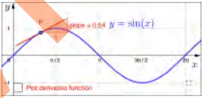
Let the first line be given by $y = m_1x + c_1$ and the second line by $y = m_2x + c_2$, then the angle between both lines is given by:

$$
\begin{aligned}
\tan(\theta) &= \frac{m_2 - m_1}{1 + m_1m_2} \\
\tan(\pi/2) &= \frac{1}{0} = \infty = \frac{m_2 - m_1}{1 + m_1m_2} \\
0 &= 1 + m_1m_2 \\
m_1m_2 &= -1
\end{aligned}
$$

Not only lines have a slope



Key idea: when the slope is zero, the curve reaches a minimum or maximum



Derivatives: how do we know the slope of an arbitrary function?

- f(x) = b (constant), the derivative is 0. $f'(x) = 0$
- When f(x) = a x, the derivative is a. $f'(x) = a$
- (sum rule) When f(x) = a x + b, the derivative is a. $f'(x) = a$
- When $f(x) = x^n$, the derivative is nx^{n-1}. $f'(x) = nx^{n-1}$
- When f(x) = cos(x), the derivative is $-sin(x)$. $f'(x) = -sin(x)$
- (product rule) $(fg)' = fg' + gf'$

Example: for $f(x) = 4x^2 + 3x$ the derivative is: $f'(x) = 8x + 3$. When deriv is zero, there is an extremum: x = -3/8

5.5 Working (lab) sessions

Four lab sessions will be organized accompanied by additional sessions “on demand”.

: a reader for every lab session is included

- Lab session 0
- Lab session 1
- Lab session 2
- Lab session 3

5.6 Additional information, slides, articles

Postgraduate Programme in 'Rehabilitation & Human Sustainable Technology'
Introduction to Technology in Physiotherapy I
Introduction to Programming: Python Installation Guide

Bart Jansen
Lecturer
bjansen@etrovub.be

Redona Brahimetaj
Teaching assistant
rbrahime@etrovub.be

Programming has become an essential skill in many fields. As a physiotherapist, you often collect a lot of data during patient assessments and treatment (outcome) sessions. Programming allows you to analyze the data more efficiently, helps to identify trends/patterns, treatment effectiveness, conduct statistical analysis, visualize results etc. Among the many-existing programming languages, Python is the most used language in data science. It offers a wide range of libraries (collection of pre-written functions to perform specific tasks) and frameworks (broader software structure that provides the foundation, conventions and tools for building entire applications) which are essential for data reading, manipulation, analysis and machine/deep learning.

In this preparatory lab session we will provide a step-by-step instruction on how to install Python (libraries) and Pycharm (an environment specifically designed to run Python code), two essential tools that will leverage your journey into programming. It is mandatory to complete the installation of both Python and Pycharm prior to the start of the first lecture. In the final section of the document, we provide supplementary information that is recommended for exploration. However, the supplementary part is optional since we will illustrate it during the lecture/exercise session.

1 Installing Python

Python is a beginner-friendly programming language. To install Python on your computer, you should follow these steps:

1. Download Python:
   - Go to the official Python website and choose the latest version[1].
   - Scroll down and select the installer for your operating system (Windows, macOS or Linux).
2. Run the installer:
   - For Windows: run the downloaded .exe file and follow the installation wizard's instructions.
   - For macOS: Open the downloaded .pkg file and follow the instructions.
   - For Linux: Open a terminal and navigate to the directory containing the downloaded .tgz file. Use the appropriate package manager to install Python (e.g., sudo apt-get install python3 for Ubuntu).
3. Check installation:
   - Open the terminal, type 'python –version' or 'python3 –version' and press Enter. Both commands should reutrn the installed Python version.

[1] Usually Python 3.x, where x is the version number.

2 Installing PyCharm

PyCharm is an integrated development environment (IDE) designed specifically for Python programming. It provides a user-friendly interface and various tools to assist in writing, debugging, and managing your Python code. To have PyCharm installed on your computer you should follow these steps:

1. Download the latest PyCharm installer from the JetBrains website. The 'Community' edition is suitable for beginners and general Python development. It is completely free, although the website might try to convince you to prefer a paid version.

2. Run the PyCharm installer to initiate the installation process and follow the instructions.

3. After completing the installation, you can launch PyCharm from your applications menu. All the read/write/edit operations on your Python code can/will be performed in the created PyCharm environment.

2.1 Creating a New Project in Pycharm

After installing PyCharm, open the application and perform the following steps to create a new project:

1. Click on the 'File' in the top menu, select 'New Project' from the dropdown manu.

2. In the 'New Project' window, you can choose where to store the project: either keep the default location or specify a new one. Write a project name, for example 'PythonBasics'.

3. Ensure that you have selected the Python interpreter. Click the 'Create' button.

4. In the project window, right-click on the project folder, hover over the 'New' in the context menu that shows up. Select 'Python File' from the submenu, enter a file name, for example 'python_basics'. Click 'Ok'.

5. In the 'python_basics.py' file that you just created, type the following code: 'print('Hello Physiotherapist')'. Do not copy the outer quotes.

6. Right click anywhere in the editor window where your code is located and select the .py script to run. Alternatively, you can also run the file from the context menu by pressing the run button.

7. The output of your Python program will be displayed in the 'Run' window at the bottom of the interface. You should see the 'Hello Physiotherapist' printed there.

3 Optional - Installing Python Libraries

Libraries (in the context of programming) are a collection of pre-written code modules, functions and data that can be used to perform specific tasks. They are designed and created to reduce the likelihood of errors and to save time and effort by providing reusable code. In this section, we will show how to install Python libraries.

1. To install Python Libraries, you will need to open a terminal (or command prompt) on your computer and use the 'pip install' command following by the name of the library you want to install, example: 'pip install numpy'.

2. You can also specify a particular version of a library to install. To do so, you should use the double equals sign followed by the version number, example: 'pip install package-name==version'. You can visit the PyPI website for the library you are interested in installing and browse the 'Release History' section to see available versions.

3. To update a library to its latest versions, you can use the ‘pip install –upgrade’ command, example: ‘pip install –upgrade package-name’

4. Some supplementary information:

   - You can create a requirements.txt file listing all the libraries and their versions, then install them all at once by typing in your terminal: ‘pip install -r requirements.txt’
   - To uninstall a library, you can use the uninstall command: ‘pip uninstall package-name’.
   - To check the installed libraries use: ‘pip list’.
   - Expect from installing Python packages from the terminal, you can also install them via Pycharm. Open the project where you want to install the Python packages. In the top menu, click on ‘File’, select ‘Preferences’ from the dropdown menu. In the ‘Preferences’ window, select your project and you will see a list of installed packages in the ‘Packages’ tab. To install a new one, click on the ‘+’ button, search for the name of the package you want to install and click the ‘Install’ button.

Helpful links: Course Notes, Python, Python Installation, Pycharm, Pycharm Installation, Project-creation Pycharm, Installing Packages, Python Packages.

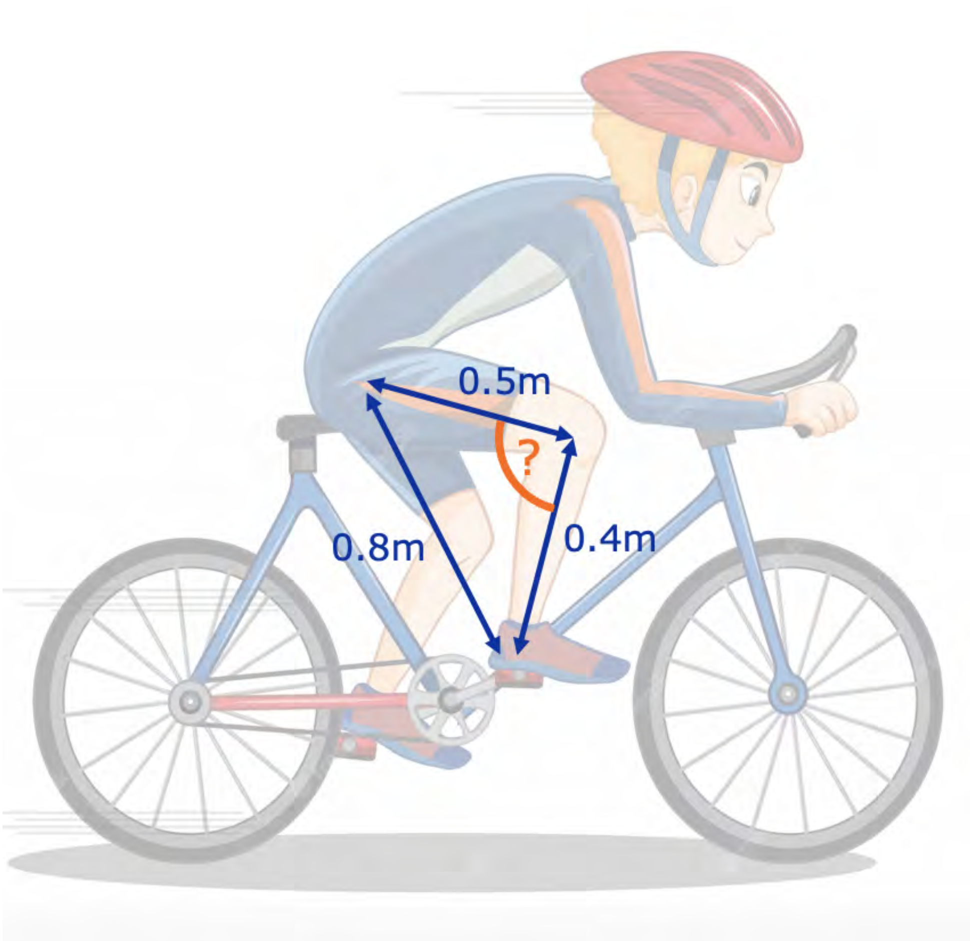
Postgraduate Programme in 'Rehabilitation & Human Sustainable Technology'
Introduction to Technology in Physiotherapy I
Exploring Python Programming - Fundamental Concepts

Bart Jansen
Lecturer
`bjansen@etrovub.be`

Redona Brahimetaj
Teaching assistant
`rbrahime@etrovub.be`

This first set of exercises practices the use and definition of: (a) variables and functions, (b) the 'if' statement and (c) the 'for' loop. As these exercises focus on these essential concepts, they naturally don't solve particularly interesting problems!

1 Numbers



1. The distance between the hip and the foot of a cyclist is measured and is equal to 0.8m. If his upper and lower leg are respectively 0.5m and 0.4m long, what is the angle of the knee? Do this with pen and paper as well as in Python.

2 Functions

1. Write a function max(x,y) that returns the largest number for two given numbers x and y. Verify its correctness by testing various input combinations.

2. Write a function that prints the following drawing:

```
*
**
***
```

3. Write a function that takes the length and width of a rectangle as input arguments and displays both its area and perimeter to the user.

4. Write a function that determines whether a given number is even or odd.

5. Write a function that calculates the body mass index of an adult and provides an interpretation (such as categorizing it as underweight, within a healthy range, indicative for morbid obesity, ...). Look up the definitions on the internet, determine the correct arguments for the function.

3 Stars and Stripes - Functions and For Loop

This series of exercises is a classic set of tasks focused on creating star patterns through drawing. These exercises are designed to help you practice in creating functions and utilizing the 'for' loop(/s). The complexity of the exercises increases progressively, so attempting to solve the final ones without mastering the earlier ones would be counterproductive.

1. Write a function that prints a row of stars (*). The function should take as arguments the number of stars to be printed. The stars should be printed on the same line in a row. For example, if you provide the number 5, the function should display a row of 5 stars like this:

   ```
   *****
   ```

   To avoid printing a new line after calling the print function, you can use print("text", end=""). Of course, this function should also work for values other than 5.

2. Write a function that prints n rows of n stars. If you provide the number 5, you get:

   ```
   *****
   *****
   *****
   *****
   *****
   ```

3. Write a function that also prints a given number of rows, but each row contains one more star. If you provide the number 5, you get:

   ```
   *
   **
   ***
   ****
   *****
   ```

4. Create a function so that the following figure is created. If you provide the number 5, you get:

```
*
**
***
****
*****
****
***
**
*
```

5. Create a function so that the following figure is created. If you provide the number 5, you get:

```
*
 *
  *
 *
*
```

6. Without trying, reason whether all these functions will also work correctly for 1 as input? What about 0? Verify.

7. Create a function so that the following figure is created. If you provide the number 5, you get:

```
  *
 *
*
 *
  *
```

8. Create a function so that the following figure is created. If you provide the number 5, you get:

```
  *|
 * |
*  |
 * |
  *|
```

9. Create a function so that the following figure is created. If you provide the number 5, you get:

```
  *|*
 * | *
*  |  *
 * | *
  *|*
```

10. Create a function so that the following figure is created. If you provide the number 5, you get:

```
  * *
 *   *
*     *
 *   *
  * *
```

11. Create a function so that for input n a diamond of n rows is created, with 2 lines of n stars on the middle row. If you provide the number 5, you get:

```
      * *
     *   *
*****     *****
     *   *
      * *
```

12. Now you've completed 11 exercises. Look back at the code from exercise 11. You probably wouldn't have been able to write this if you hadn't done the first 10 exercises. Think of other complex patterns to draw. Consider how you split them into simpler steps and code from simple to complex.

Helpful links: Course Notes, Python, Math Library, Functions, 'If' Condition(s), 'For' Loops

Postgraduate Programme in 'Rehabilitation & Human Sustainable Technology'
Introduction to Technology in Physiotherapy I
Exploring Python Programming - Lists and Fundamental Concepts

Bart Jansen
Lecturer
bjansen@etrovub.be

Redona Brahimetaj
Teaching assistant
rbrahime@etrovub.be

In this lab session, you will work with some basic programming concepts and operations such as: (a) creating and using variables; (b) changing the behaviour of your code by using the conditional 'if' statements; (c) explore the concept of 'for' loops; (d) gain proficiency in working with lists and strings; (e) as you progress through the exercises you will improve your programming skills in creating/using functions.

1 Lists - Operations

1. Test the basic operations from today's slide set!

2 Lists - Exercises

1. Create a function that prints each element of a given list.

2. Create a function that searches for the largest element in a list. Ensure that the function returns the position of the largest element. Call the function and print: "The largest element is X and is at position Y", for the correct X and Y.

3. Create a function that, in a list of numbers, replaces every number less than 10 with 0.

4. Create a function that takes 2 lists as arguments (assuming that the lists are of the same length and only contain numbers). The function should return a new list in which each element is the sum of the corresponding elements in the 2 lists.

5. Create a function that returns all elements that appear in 2 given lists.

6. A physiotherapy clinic keeps records of patient progress (in a simplistic form). The clinic maintains two lists: 'patient_sessions' (used to store the number of sessions each patient has attended) and 'patient_assessment' (to store the assessment scores of the physical condition of each patient). Create a Python function (or a set of functions) to analyze and determine if a patient has made progress or experienced no improvements in their physical condition.

7. A physiotherapy clinic occasionally provides special non-slippery socks to its patients during different therapy sessions. The clinic maintains a stock of these socks in different sizes (for convenience in sizes 0 through 10). The total stock of each size of socks is maintained by a list. The list's positions correspond to the sock sizes (position 0 for size 0, and so on). Create several functions that take the stock list as an argument (+ possibly other arguments) and allow for stock management: printing the total stock, finding out how many socks of a certain size there are, adjusting the stock when a patient has used a certain sock, and finally a function that prints which socks urgently need to be restocked, namely all those socks of which there are fewer than 10 available.

8. We simplified the task by assuming that the socks' sizes are from 0 to 10, which is not the case. How can you solve exercise 7 for real sizes?

3 Modelling the queues at the emergency care department.

1. The elements in a list do not necessarily have to be numbers. Create a list where each element is a string of 2 or 3 letters. The first letter must be a consonant, and the last one a vowel (e.g. bra, du, tee, la, flo, har, ba...).

2. Create a function makePatient(), which generates a name for a new patient by concatenating 2 or 3 random syllables from the list of exercise 1. Use the "randrange" function from the random module for this.

3. In computer science, a queue is an essential data structure, characterized by two core operations: "push(element)" which appends an element to the end of the queue, and "pop()" which extracts an element from the front.

   Create the functions makeQueue(), push(queue, element), and pop() to manage a queue using a list.

4. Evaluate the function by simulating a queue scenario at the emergency care department: Use a 'for' loop that, at each iteration, employs a random dice roll to determine whether a new patient joins the queue or whether a patient is served.Always print sufficient information.

5. Inevitably there will be a situation where you want to help a patient, but the queue is empty. Ensure that your code does not crash in such a scenario!

6. The above is of course not a proper simulation of the queue at the emergency care department as the waiting time in the above exercise is not influenced by the urgency of the problem. Therefore, extend the simulation as follows: upon arrival at the emergency care dept, a patient is assigned the label "urgent" or "not urgent". Both patient types are added to a separate queue. Whenever a new patient can be served, patients in the "urgent" queue have absolute priority. So, only when the "urgent" queue is empty, a patient from the non-empty queue is served.

7. The above extended version is of course still not realistic at all. Provide any improvement to the scenario.

Useful links: Python Lists, Functions, 'If' Condition(s), 'For' Loops,Random Module

Postgraduate Programme in 'Rehabilitation & Human Sustainable Technology'
Introduction to Technology in Physiotherapy I
Exploring Python Programming - NumPy library and Vizualizations

Bart Jansen
Lecturer
bjansen@etrovub.be

Redona Brahimetaj
Teaching assistant
rbrahime@etrovub.be

In this lab session you will be working with NumPy, a fundamental Python library used for scientific computing. It is memory/super-efficient and allows to perform simple and complex calculations (faster than Python lists) on large datasets. Raw data can be challenging to understand, no matter how extensive and accurate it can be. Visualizations are very important to understand insights hidden within the data. In addition to exploring NumPy, in this lab session you will create different plots using 'Matplotlib' plotting library.

1 NumPy - Operations

1. Create a NumPy array to store the names of five patients.
2. Create another array that store the therapy session duration (in minutes) for each patient. Create a function that calculates the average duration per patient.
3. Create: (a) two arrays that store the patient's height (in cm) and weight (in kg); and (b) a function that computes the body mass index (BMI) for each patient.
   - Identify (and print) if there are patients with a BMI below 18.5 and/or above 25 (indicating under/over-weight).
   - Modify your function to provide the BMI category for each patient.
4. Monitoring daily physical activity is important for rehabilitation. Create a function that generates a random[1] array representing daily step counts for seven days for all of the five patients defined in the first exercise. Calculate the total weekly step count per each patient. Identify the least and the most active patient.
   - Does your function handle cases where multiple patients have the same total weekly steps? If not, please make the necessary adjustments to handle such scenario.
5. Convert the lists defined in exercise 2.6 (lab session 2) into numpy arrays. Calculate the Pearson correlation coefficient between 'patient_sessions' and 'patient_assessment'. Interpret the results.
6. Create two arrays that represent the pain scores on the first and last day of the physiotherapy session. The pain scores should be on a scale 0-10 and you have to generate the values yourself. Perform a paired t-test to determine if there exists a statistically significant difference in pain scores before and after the completion of the physiotherapy sessions.
7. Imagine you have a n-dimensional dataset containing knee-joint angle measurements for a group of robust and frail subjects. Each row represents a subject, and each column represents a different joint angle at various time points over a 90-second interval. Additionally, at the end of each row, you have a binary classification label indicating whether the subject is robust (0) or frail (1).
   - Create a 2D NumPy array that represents this dataset. Generate random joint-angle values that fall within a reasonable range (for frail/robust cases).
   - Calculate the mean joint angle for each patient during the mid-60 second period.
   - Find the time point where the patients achieved the highest range of motion. Is the time peak significantly different from the robust and frail cases?

[1] In real-world scenarios, you will be reading files that contain real patient's daily step count for specific duration. In this exercise, nevertheless we are replacing the real-data with random-generated ones.

2 Visualizations - Plotting in Python

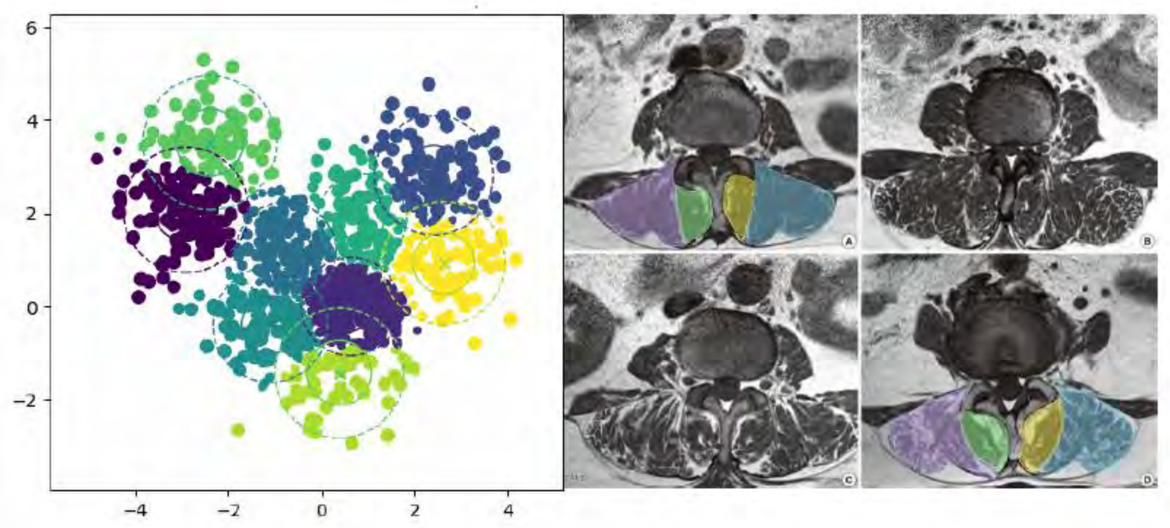
1. Draw a simple lineplot to represent the change in a patient's heart rate over time (2 minutes). Use randomly generated data (that make sense) for heart rate values.

2. Generate a histogram plot showing the distribution of the BMI values computed (in exercise 1.3) among the patients. Add an additional array to store the patient gender and generate again same histogram plot but now grouping the BMI per gender type.

Useful links: Python, NumPy, Numpy Array Creation, Functions, 'If' Condition(s), 'For' Loops,Random Module, Pearson Correlation, T-test, Vizualizations - MatPlot Library, Line plots, Histogram plot

5.7 Application of Computer vision models

Guest lecture by Eddo Wesselink on Application of Computer vision models for the automatic quantification of lumbar paraspinal muscle health



: ppt via the guest lecturer

5.7.1 OBJECTIVE

5.7.1.1 Session 1

5.7.1.1.1 Learn why we need computer-vision to improve our understanding of lumbar paraspinal muscle health decline

5.7.1.1.2 Learn what computer vision is

5.7.1.1.3 Learn how to interpret computer-vision performance

Questions can be asked between the sub-objectives

5.7.1.2 Session 2 (Practical)

Use a pre-trained computer-vision model to segment the lumbar paraspinal muscles

To do for the practical session

1. Install the required python packages with the following commands
   - # pip install os
   - # pip install glob,

- # pip install torch
- # pip install monai
- # pip install nibabel
- # pip install pandas
- # pip install numpy

2. Look into the shared documents and
   - Read the segmentation metrics document
   - Watch the video for using the programming codes (see learning platform)

5.8 Additional information, slides, articles

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

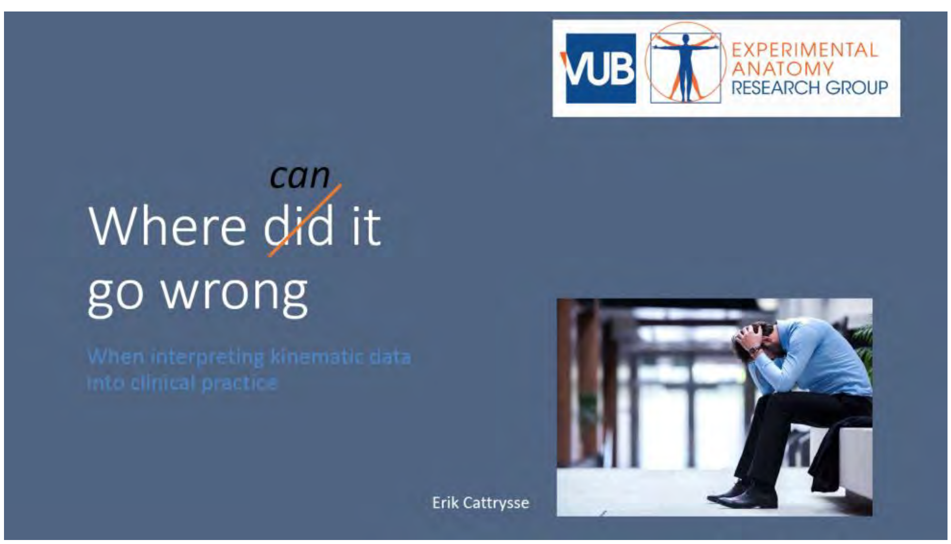
Part 3 Movement registration

1 Introduction to artrokinematics

How:

- online synchronous (live) (see online schedule)
- followed by online (live) working sessions (see online schedule)
- and on campus working sessions (see online schedule intensive week)

1.1 Joint-kinematics Analysis, from quantity to quality



: Joint-kinematics Analysis, from quantity to quality

1.2 Additional information, slides, articles

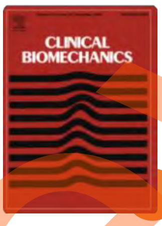
Joint-kinematics Analysis, from quantity to quality

Erik Cattrysse

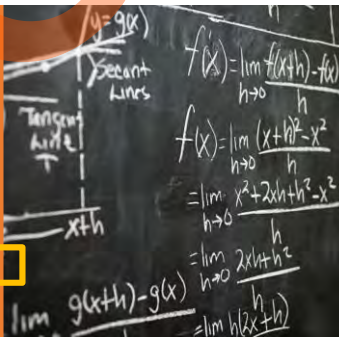
Where ~~did~~ *can* it go wrong

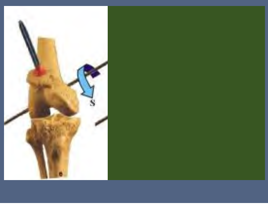
When interpreting kinematic data into clinical practice

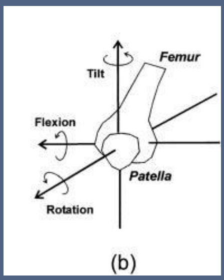
Erik Cattrysse

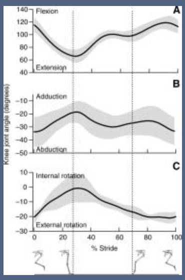


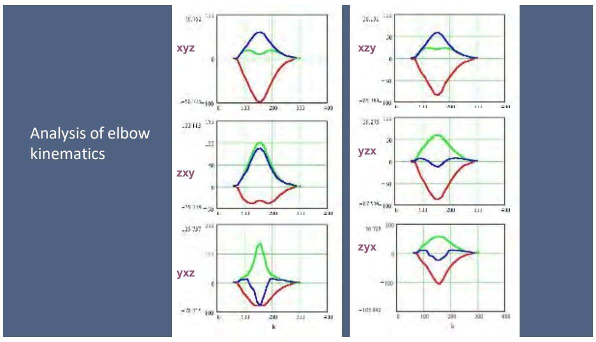
Some examples

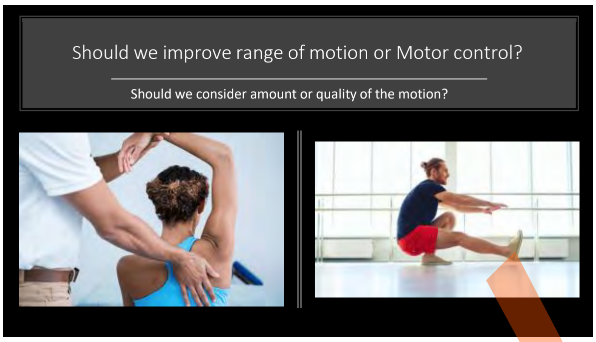




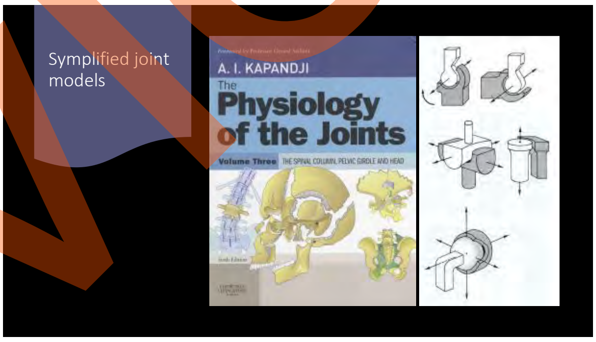


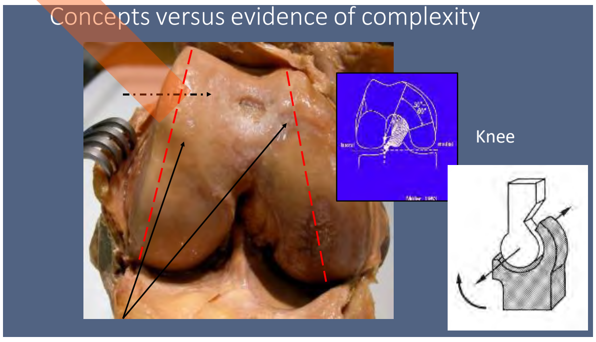


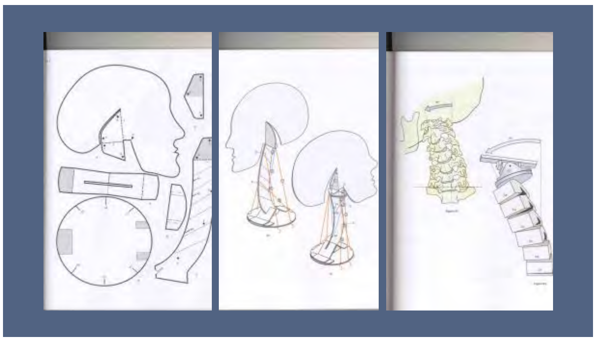

Analysis of elbow kinematics
xyz
zxy
yxz
xzy
yzx
zyx

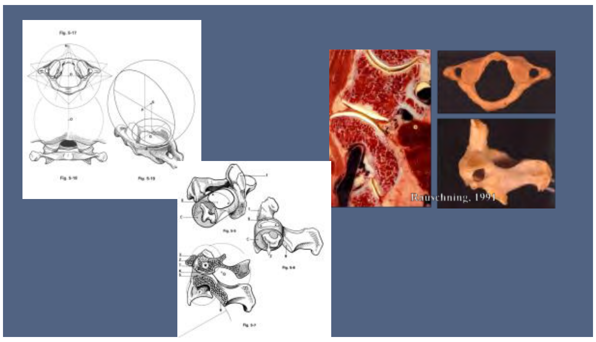

Should we improve range of motion or Motor control?
Should we consider amount or quality of the motion?

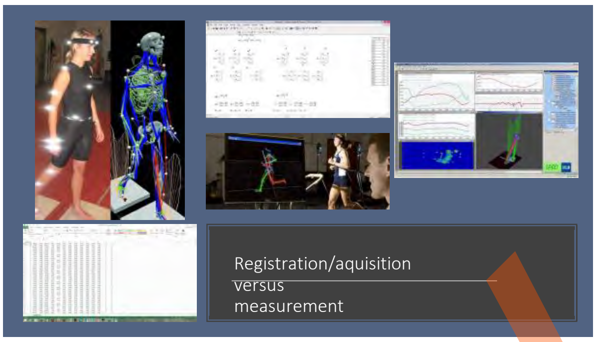

Let's think about it ....


Symplified joint models
A. I. KAPANDJI
The Physiology of the Joints
Volume Three


Concepts versus evidence of complexity
Knee

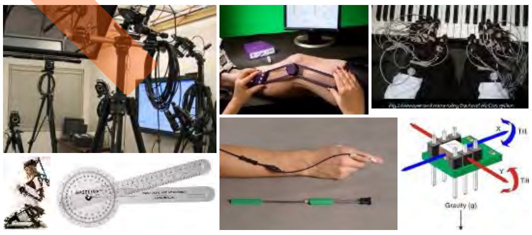



Rauschning, 1991


Registration/aquisition
versus
measurement


Biomechanical methods and methodologies
Registration
What do we register?
How do we register?
Analysis
What do we analyse?
How do we analyse?
Why do we analyse?

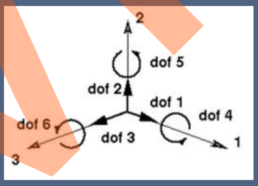

What do we register?
• E. Muybridge




How do we register?
Static
Dynamic

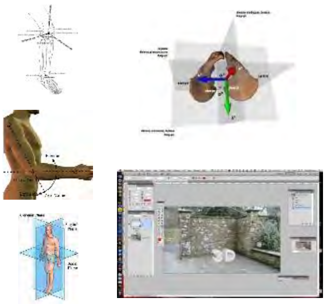
Why do we register?

Interpretation of intra-articular motion based on 6Dof-kinematics



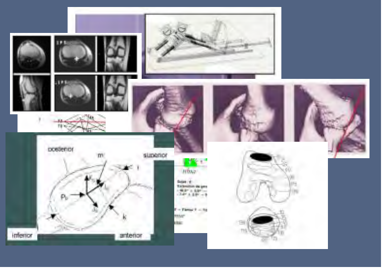
Reference frames

- Absolute / technical
- (functional) Anatomical
- Local
- Bone embedded

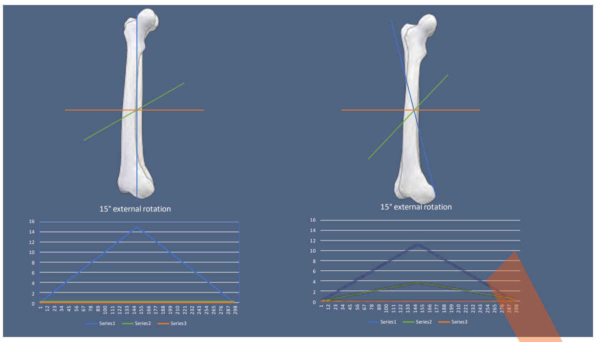


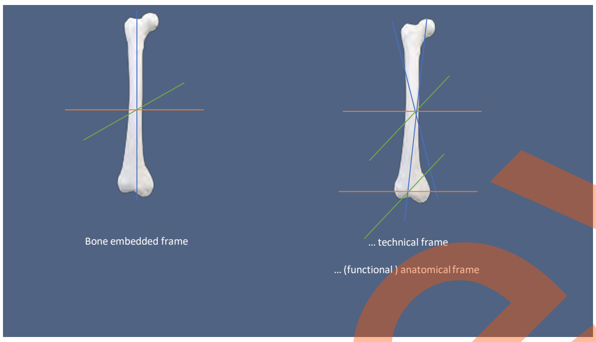
Intra-articular kinematics

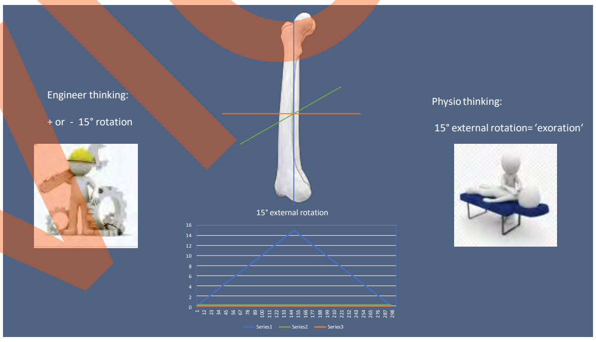
- Local bone embedded reference frames
- Contact areal analysis

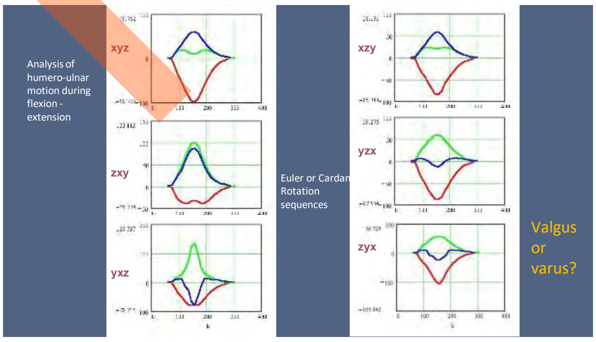


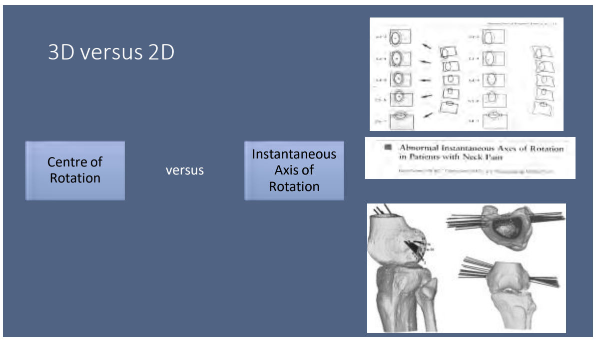

'Angles
versus
Angles'


15° external rotation
15° external rotation
Series1
Series2
Series3

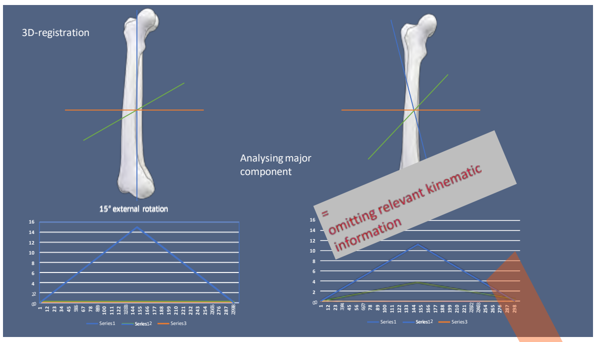

Bone embedded frame
... technical frame
... (functional) anatomical frame

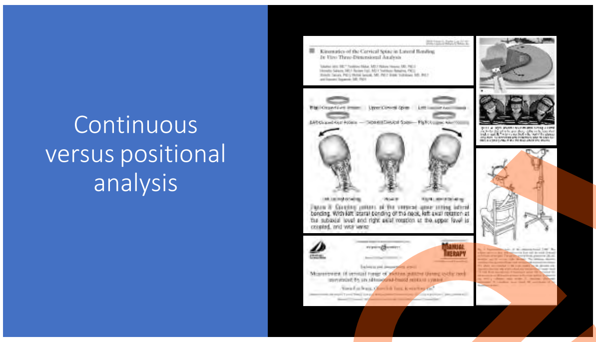

Engineer thinking:
+ or - 15° rotation
15° external rotation
Physio thinking:
15° external rotation= 'exoration'

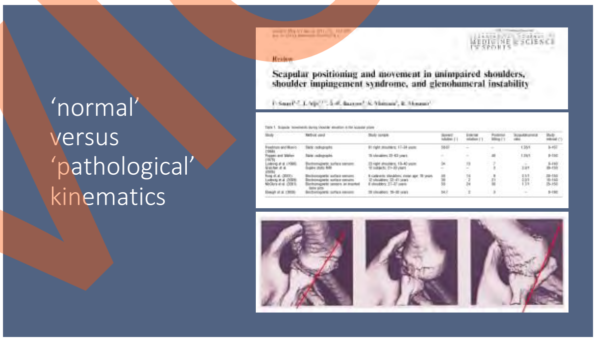

Analysis of humero-ulnar motion during flexion - extension
xyz
zxy
yxz
Euler or Cardan Rotation sequences
xzy
yzx
zyx
Valgus or varus?

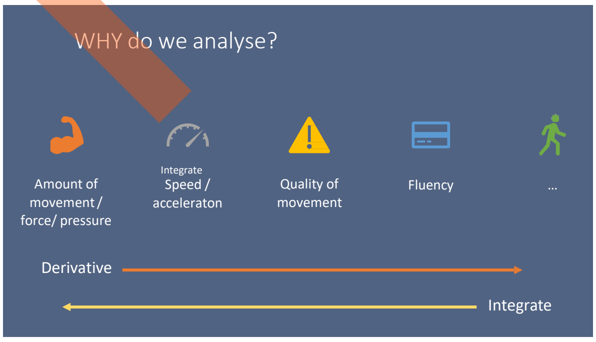

3D versus 2D
Centre of Rotation
versus
Instantaneous Axis of Rotation
Abnormal Instantaneous Axis of Rotation in Patients with Neck Pain

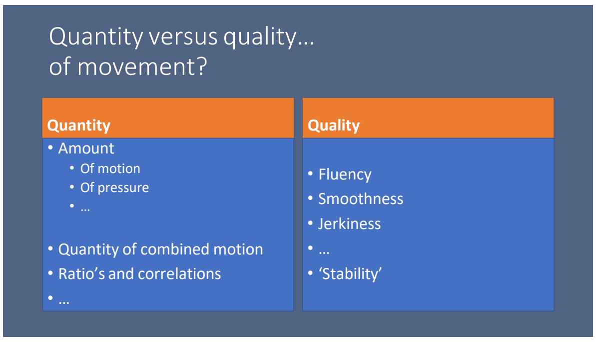

3D versus 3 x 2D
•3D ≠ 3 x 2D

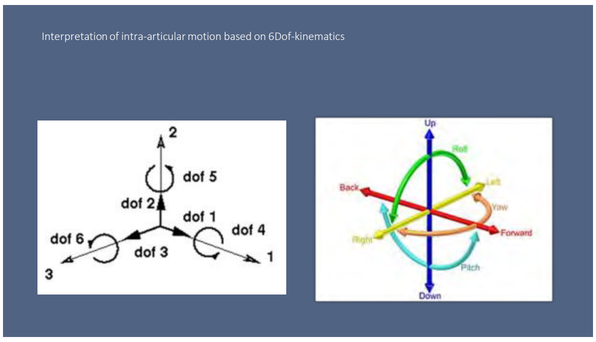

3D-registration
15° external rotation
Analysing major component
= omitting relevant kinematic information

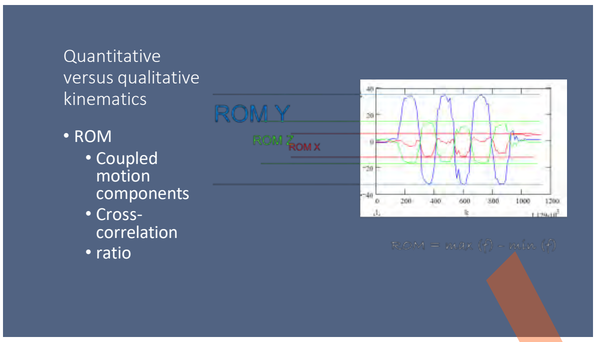

Continuous versus positional analysis

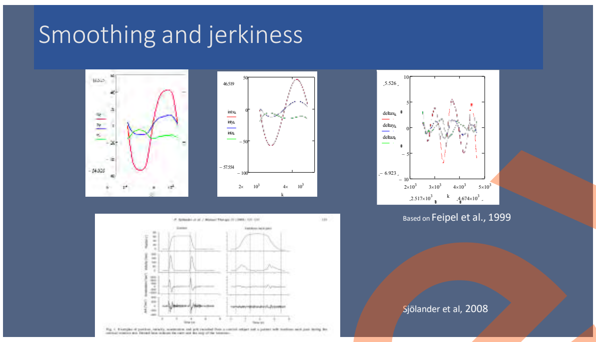

'normal' versus 'pathological' kinematics
Scapular positioning and movement in unimpaired shoulders, shoulder impingement syndrome, and glenohumeral instability

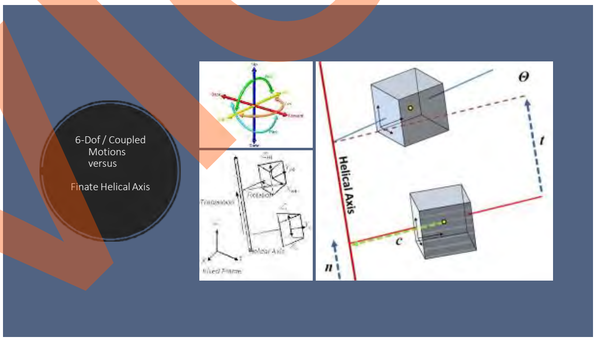

WHY do we analyse?
Amount of movement / force/ pressure
Integrate
Speed / acceleraton
Quality of movement
Fluency
...
Derivative
Integrate

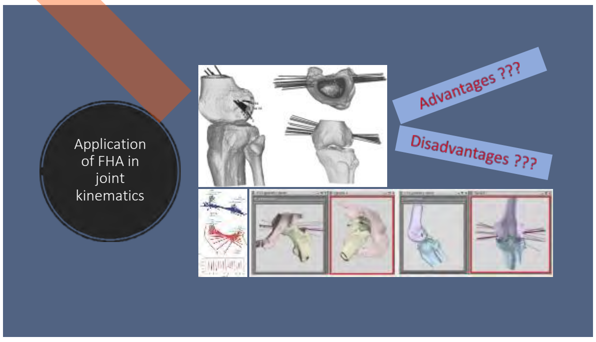

Quantity versus quality... of movement?
Quantity
• Amount
• Of motion
• Of pressure
• ...
• Quantity of combined motion
• Ratio's and correlations
• ...
Quality
• Fluency
• Smoothness
• Jerkiness
• ...
• 'Stability'

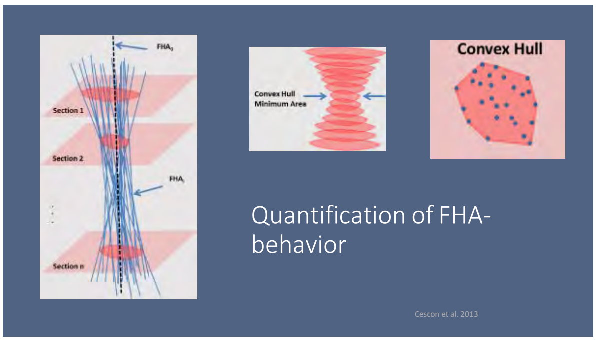

Interpretation of intra-articular motion based on 6Dof-kinematics
2
dof 5
dof 2
dof 1
dof 6
dof 4
dof 3
3
1
Up
Roll
Back
Left
Yaw
Right
Forward
Pitch
Down


Quantitative versus qualitative kinematics
• ROM
• Coupled motion components
• Cross-correlation
• ratio
ROM Y
ROM Z
ROM X

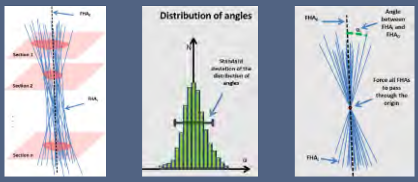

Smoothing and jerkiness
Based on Feipel et al., 1999
Sjölander et al, 2008

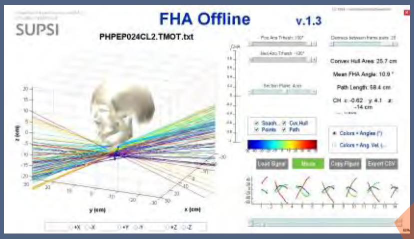

6-Dof / Coupled Motions versus
Finate Helical Axis
Helical Axis
θ
t
c
n


Application of FHA in joint kinematics
Advantages ???
Disadvantages ???


Section 1
Section 2
Section n
Convex Hull Minimum Area
Convex Hull
Quantification of FHA-behavior
Cescon et al. 2013

Cescon et al. 2013



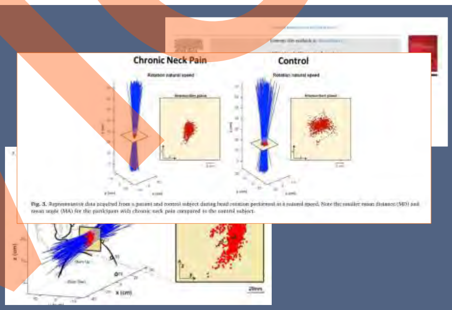


FHA-analysis

Previous studies using FHA in cervical kinematics

- Cescon, C., et al. (2014). "Methodological analysis of finite helical axis behavior in cervical kinematics." Journal of Electromyography and Kinesiology **24**(5): 628-635.
- Barbero, M., et al. (2017). "Can parameters of the helical axis be measured reliably during active cervical movements?" Musculoskeletal Science and Practice **27**: 150-154.
- Asultan, F., et al. (2019). "Variability of the helical axis during active cervical movements in people
- with chronic neck pain." Clinical Biomechanics **62**: 50-57.

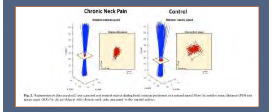
52



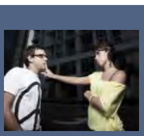
53


ORIGINAL ARTICLE

Cervical kinematics estimated by finite helical axis behaviour differs in patients with neck related problems as compared to healthy controls



Is there a need for....

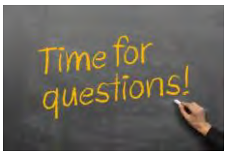


- a paradigm shift
- ...

- better applied biomechanics education in physiotherapy ?
- more basic medical science in engineering education ?

- an *engineering physiotherapist* or *physiotherapeutical engineer* ?
  - Post Graduate course in *Health Technology and sustainable human rehabilitation*

?



VUB EXPERIMENTAL ANATOMY RESEARCH GROUP

2 From coordinates to angles

2.1 Chapter 1: Definition of a local coordinate frame

- Types of reference frames
- Why 3 points in space?
- : Definition of a local coordinate frame

2.2 Rotating a reference frame

- What are Cardan angles? (TPS)
- What are Euler Angles? (TPS)
- : Rotating a reference frame

2.3 Applying rotation sequences in angle calculation

- : Rotation sequences
- Excercise: angle calculation

2.4 Additional information, slides, articles

Artrokinematics

"From coordinates to angles"

E.Cattrysse

topics

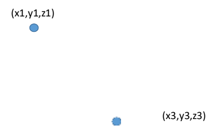
- Definition of a local coordinate frame
  - Types of reference frames
  - Why 3 points in space?
  - Ppt: Definition of a local coordinate frame
- Rotating a reference frame
  - What are Cardan angles? (TPS)
  - What are Euler Angles? (TPS)
  - Ppt: Rotating a reference frame
- Applying rotation sequences in angle calculation
  - Ppt: Rotation sequences
  - Excercise: angle calculation

Definition of a local coordinate frame

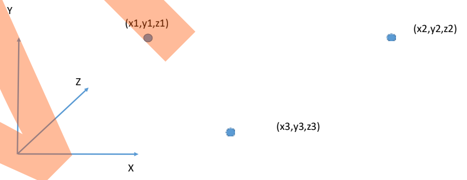
E. Cattrysse

Three non-linear points in space define a local coordinate frame

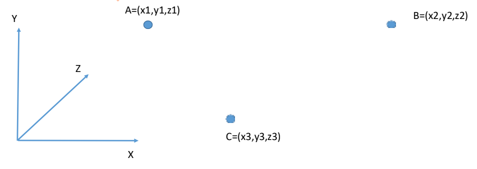
The points are defined by their coordinates



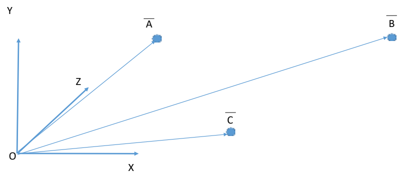
These coordinates are expressed with respect to the general coordinate frame



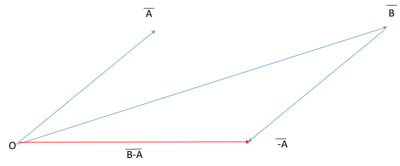
They represent three vectors that have their origin at the origin/center of the general reference frame and point towards point A, B and C



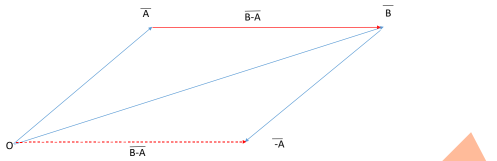
They can thus be represented by their spatial vector description

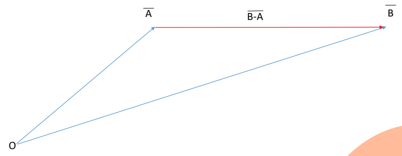


Using simple vector distraction, it can be shown that the line connecting A and B can be defined as a new vector B-A

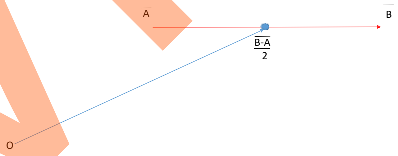


Using simple vector distraction, it can be shown that the line connecting A and B can be defined as a new vector B-A

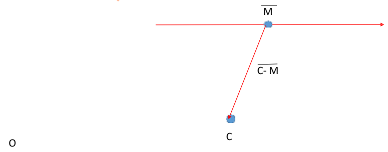




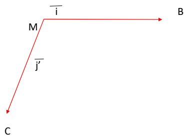
The midpoint of the line B-A can be definied as the vector: M= $\frac{\overline{B-A}}{2}$



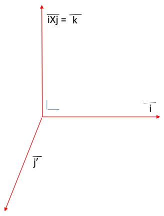
We can now define a vector from point C to M



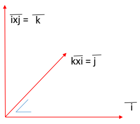
Let's denote the vector B-M as i and the vector C-M as j'



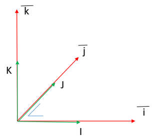
The vector product: ixj
a vector product results in a vector perpendicular to the plane defined by the two vectors (= norm vector)



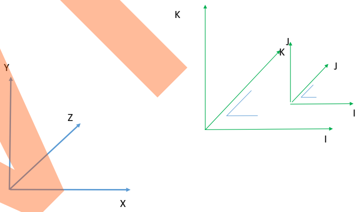
To make all vectors perpendicular we add vector product: kxi=j which is situated in the same plane as J' but this time perpendicular to the plane defined by K and I



By deviding the vectors by their absolute value("length") we create the unit vectors I, J and K



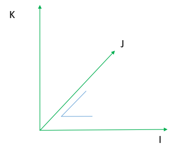
We have now created a local orthogonal reference frame IJK (= local frame) relative to the general reference frame XYZ



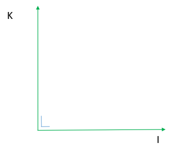
Rotating a reference frame in 3D-space:
How to define the rotation matrix?

E.Cattrysse

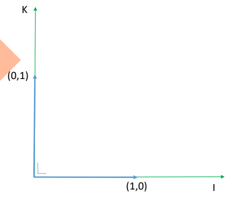
Lets start with a(n) (local) orthogonal reference frame



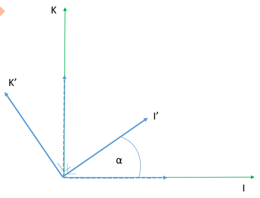
Knowing that the principles of creating a rotation matrix in 3D space are similar but slightly more complex to 2D, let's start with 2D



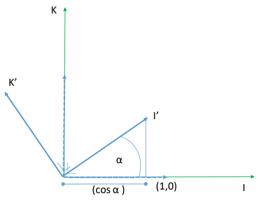
In this frame we consider the two unit vectors on the axes I and K with coordinates (1,0) and (0,1)



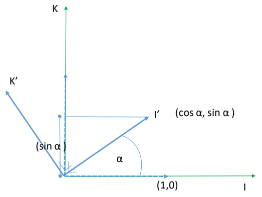
Next the frame can be rotated around it's center O abouth an angle α



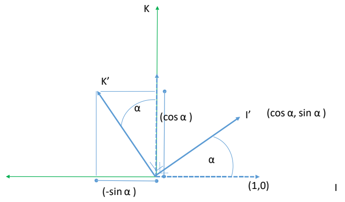
The unit vector I becomes I' and its new coordinates can be defined using cosine and sinus functions



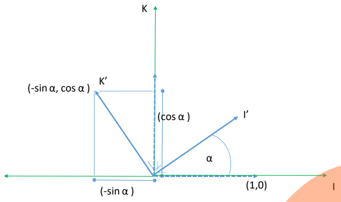
The coordinates of I’ are (cos α , sin α)



Similarly it can be demonstrated that the coordinates of K’ are (-sin α , cos α) as the angle between K and K’ is the same as the angle α between I and I’



The coordinates of the rotated vector K’ are (-sin α , cos α)



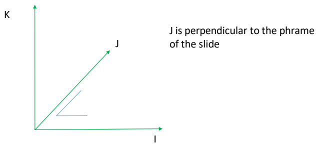
The original frame was defined by its two unit vectors incorportated in the rotation matrix:

$$R= \begin{bmatrix} 1,0 \\ 0,1 \end{bmatrix}$$

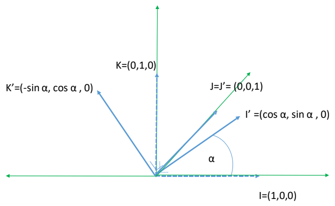
After rotation abouth α its rotation matrix has now become:

$$R^{\alpha}= \begin{bmatrix} \cos\alpha, -\sin\alpha \\ \sin\alpha, \cos\alpha \end{bmatrix}$$

We can now expand this concept to 3D-space, introducing again the third axis J



Rotating the frame abouth α around the axis J will result in the following



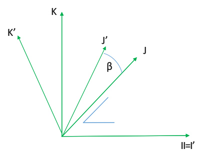
In 3D-space the original frame was defined by its three unit vectors incorportated in the rotation matrix:

$$R= \begin{bmatrix} 1,0,0 \\ 0,1,0 \\ 0,0,1 \end{bmatrix}$$

After rotation abouth α its rotation matrix has become:

$$R^{\alpha}= \begin{bmatrix} \cos\alpha, & -\sin\alpha, & 0 \\ \sin\alpha, & \cos\alpha, & 0 \\ 0 & , 0 & , 1 \end{bmatrix}$$

However, if we now consider a rotation of the original fram IJK abouth an angle β around the axis I (=rotation on the JK-plane)



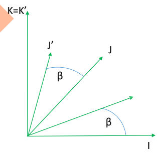
In 3D-space the original frame rotation matrix R:

$$R= \begin{bmatrix} 1,0,0 \\ 0,1,0 \\ 0,0,1 \end{bmatrix}$$

After rotation abouth β its rotation matrix has become:

$$R^{\beta}= \begin{bmatrix} 1 & , 0 & , 0 \\ 0 & , \cos\beta & , -\sin\beta \\ 0 & , \sin\beta & , \cos\beta \end{bmatrix}$$

Similarly we can consider a third kind of rotation of the original fram IJK abouth an angle γ around the axis K in the IJ-plane



In 3D-space the original frame rotation matrix R:

$$R= \begin{bmatrix} 1,0,0 \\ 0,1,0 \\ 0,0,1 \end{bmatrix}$$

After rotation abouth γ its rotation matrix has become:

$$R^{\gamma}= \begin{bmatrix} \cos\gamma & , 0, & -\sin\gamma \\ 0 & , 1, & 0 \\ \sin\gamma & , 0, & \cos\gamma \end{bmatrix}$$

Complex rotations in 3D space can be considered as a combination of rotations around the three axes of the reference frame ; the result however will be sequence dependend

$$R^{\alpha}= \begin{bmatrix} \cos\alpha, & -\sin\alpha, & 0 \\ \sin\alpha, & \cos\alpha, & 0 \\ 0 & , 0 & , 1 \end{bmatrix} \quad R^{\beta}= \begin{bmatrix} 1 & , 0 & , 0 \\ 0 & , \cos\beta & , -\sin\beta \\ 0 & , \sin\beta, & \cos\beta \end{bmatrix} \quad R^{\gamma}= \begin{bmatrix} \cos\gamma & , 0, & -\sin\gamma \\ 0 & , 1, & 0 \\ \sin\gamma & , 0, & \cos\gamma \end{bmatrix}$$

Rotation sequences

Cardan versus Euler Angles

E.Cattrysse

Complex rotations in 3D space can be considered as a combination of rotations around the three axes of the reference frame ; the result however will be sequence dependend

$$R^{\alpha}= \begin{bmatrix} \cos\alpha, & -\sin\alpha, & 0 \\ \sin\alpha, & \cos\alpha, & 0 \\ 0 & , 0 & , 1 \end{bmatrix} \quad R^{\beta}= \begin{bmatrix} 1 & , 0 & , 0 \\ 0 & , \cos\beta & , -\sin\beta \\ 0 & , \sin\beta & , \cos\beta \end{bmatrix} \quad R^{\gamma}= \begin{bmatrix} \cos\gamma & , 0, & -\sin\gamma \\ 0 & , 1, & 0 \\ \sin\gamma & , 0 & , \cos\gamma \end{bmatrix}$$

Rotation conventions

- Cardan Angles vs Euler Angles:see https://en.wikipedia.org/wiki/Euler_angles
- Angles can be considered as a sequence of rotations around the three basic axes of the frame (IJK will be considered as XYZ in the following)
  - The sequences of rotations can be considered as rotations around each axis in it's original/fixed position (Rx, Ry, Rz)
    - = Extrinsic rotation
  - Or around their newly intermediate positions (Rx, Rx', Rx'', Ry , Ry', Ry'', Rz, Rz', Rz'')
    - = Intrinsic rotations
- Euler Angles combine three rotation around two out of three axes
- Cardan Angles combine three rotations around all three axes

Euler angle conventions

- There are six possibilities of choosing the rotation axes for proper Euler angles. In all of them, the first and third rotation axes are the same. The six possible sequences are:
- *z-x′-z″* (intrinsic rotations) or *z-x-z* (extrinsic rotations)
- *x-y′-x″* (intrinsic rotations) or *x-y-x* (extrinsic rotations)
- *y-z′-y″* (intrinsic rotations) or *y-z-y* (extrinsic rotations)
- *z-y′-z″* (intrinsic rotations) or *z-y-z* (extrinsic rotations)
- *x-z′-x″* (intrinsic rotations) or *x-z-x* (extrinsic rotations)
- *y-x′-y″* (intrinsic rotations) or *y-x-y* (extrinsic rotations)

Cardan angle conventions (= Tait-Bryan angles)

- There are six possibilities of choosing the rotation axes for Tait–Bryan angles. The six possible sequences are:
- *x-y′-z″* (intrinsic rotations) or *x-y-z* (extrinsic rotations)
- *y-z′-x″* (intrinsic rotations) or *y-z-x* (extrinsic rotations)
- *z-x′-y″* (intrinsic rotations) or *z-x-y* (extrinsic rotations)
- *x-z′-y″* (intrinsic rotations) or *x-z-y* (extrinsic rotations)
- *z-y′-x″* (intrinsic rotations) or *z-y-x* (extrinsic rotations):
  - the intrinsic rotations are known as: yaw, pitch and roll
- *y-x′-z″* (intrinsic rotations) or *y-x-z* (extrinsic rotations)

How do differend sequences lead to different rotation matrices and as a consequence to differend angles

Principles of Matrix multiplication

Application to 3D-space

Remember from simple matrix multiplication to multiplacate each row of A with each column of B

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} x \\ y \\ z \end{pmatrix},$$

their matrix product is:

$$\mathbf{AB} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax+by+cz \\ px+qy+rz \\ ux+vy+wz \end{pmatrix},$$

Remember from matrix multiplication:

A.B ≠ B.A

If

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} \alpha & \beta & \gamma \\ \lambda & \mu & \nu \\ \rho & \sigma & \tau \end{pmatrix},$$

their matrix products are:

$$\mathbf{AB} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} \alpha & \beta & \gamma \\ \lambda & \mu & \nu \\ \rho & \sigma & \tau \end{pmatrix} = \begin{pmatrix} a\alpha+b\lambda+c\rho & a\beta+b\mu+c\sigma & a\gamma+b\nu+c\tau \\ p\alpha+q\lambda+r\rho & p\beta+q\mu+r\sigma & p\gamma+q\nu+r\tau \\ u\alpha+v\lambda+w\rho & u\beta+v\mu+w\sigma & u\gamma+v\nu+w\tau \end{pmatrix},$$

and

$$\mathbf{BA} = \begin{pmatrix} \alpha & \beta & \gamma \\ \lambda & \mu & \nu \\ \rho & \sigma & \tau \end{pmatrix} \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} = \begin{pmatrix} \alpha a+\beta p+\gamma u & \alpha b+\beta q+\gamma v & \alpha c+\beta r+\gamma w \\ \lambda a+\mu p+\nu u & \lambda b+\mu q+\nu v & \lambda c+\mu r+\nu w \\ \rho a+\sigma p+\tau u & \rho b+\sigma q+\tau v & \rho c+\sigma r+\tau w \end{pmatrix}.$$

Let's consider again the matrices Rα and Rβ

$$R^{\alpha} = \begin{bmatrix} \cos\alpha, & -\sin\alpha, & 0 \\ \sin\alpha, & \cos\alpha, & 0 \\ 0, & 0, & 1 \end{bmatrix} \quad R^{\beta} = \begin{bmatrix} 1, & 0, & 0 \\ 0, & \cos\beta, & -\sin\beta \\ 0, & \sin\beta, & \cos\beta \end{bmatrix}$$

$R^{\alpha} * R^{\beta} = ?$

$$R^{\alpha} = \begin{bmatrix} \cos\alpha, & -\sin\alpha, & 0 \\ \sin\alpha, & \cos\alpha, & 0 \\ 0, & 0, & 1 \end{bmatrix} \quad R^{\beta} = \begin{bmatrix} 1, & 0, & 0 \\ 0, & \cos\beta, & -\sin\beta \\ 0, & \sin\beta, & \cos\beta \end{bmatrix}$$

$$R^{\alpha} * R^{\beta} = \begin{bmatrix} \cos\alpha+0+0, & 0+(-\sin\alpha.\cos\beta)+0, & 0+(-\sin\alpha.-\sin\beta)+0 \\ \sin\alpha+0+0, & 0+(\cos\alpha.\cos\beta)+0, & 0+(\cos\alpha.-\sin\beta)+0 \\ 0+0+0, & 0+0+\sin\beta, & 0+0+\cos\beta \end{bmatrix}$$

$$R^{\alpha} = \begin{bmatrix} \cos\alpha, & \sin\alpha, & 0 \\ -\sin\alpha, & \cos\alpha, & 0 \\ 0, & 0, & 1 \end{bmatrix} \quad R^{\beta} = \begin{bmatrix} 1, & 0, & 0 \\ 0, & \cos\beta, & \sin\beta \\ 0, & -\sin\beta, & \cos\beta \end{bmatrix}$$

Simplified written as:

$$R^{\beta\alpha} = \begin{bmatrix} \cos\alpha, & -\sin\alpha.\cos\beta, & -\sin\alpha.-\sin\beta \\ \sin\alpha, & \cos\alpha.\cos\beta, & \cos\alpha.-\sin\beta \\ 0, & \sin\beta, & \cos\beta \end{bmatrix}$$

Multiplying the new Matrix $R^{\beta\alpha}$ a second time with R^{γ} wil result in the complex matrix $R^{\gamma\beta\alpha}$

$$R^{\beta\alpha} = \begin{bmatrix} \cos\alpha, & -\sin\alpha.\cos\beta, & -\sin\alpha.-\sin\beta \\ \sin\alpha, & \cos\alpha.\cos\beta, & \cos\alpha.-\sin\beta \\ 0, & \sin\beta, & \cos\beta \end{bmatrix} \quad R^{\gamma} = \begin{bmatrix} \cos\gamma, & 0, & -\sin\gamma \\ 0, & 1, & 0 \\ \sin\gamma, & 0, & \cos\gamma \end{bmatrix}$$

$$R^{\gamma\beta\alpha} = \begin{bmatrix} \cos\alpha.\cos\gamma+(-\sin\alpha.-\sin\beta.\sin\gamma), & -\sin\alpha.\cos\beta, & \cos\alpha.-\sin\gamma+-\sin\alpha.-\sin\beta.\cos\gamma \\ \sin\alpha.\cos\gamma+\cos\alpha.-\sin\beta.\sin\gamma, & \cos\alpha.\cos\beta, & \sin\alpha.-\sin\gamma+\cos\alpha.-\sin\beta.\cos\gamma \\ \cos\beta.\sin\gamma, & \sin\beta, & \cos\beta.\cos\gamma \end{bmatrix}$$

Distracting angles from this matrix is based on the inversed cos and sin functions using Arccos and Arcsin

$$R^{\gamma\beta\alpha} = \begin{bmatrix} \cos\alpha.\cos\gamma+(-\sin\alpha.-\sin\beta.\sin\gamma) & , -\sin\alpha.\cos\beta & , \cos\alpha.-\sin\gamma+-\sin\alpha.-\sin\beta.\cos\gamma \\ \sin\alpha.\cos\gamma+\cos\alpha.-\sin\beta.-\sin\gamma & , \cos\alpha.\cos\beta & , \sin\alpha.-\sin\gamma+\cos\alpha.-\sin\beta.\cos\gamma \\ \cos\beta.\sin\gamma & , \sin\beta & , \cos\beta.\cos\gamma \end{bmatrix}$$

β can be computed from sin β using arcsin= β
Once β is known, α can for instance be calculated from (cos α. cosβ)
And γ can for instance be calculated from (cosβ .cos γ) as β is already known

Cardan angles calculation

From the previous it may become clear that changing the sequences will results in differend complex matrices for which

$R^{\gamma\beta\alpha} \neq R^{\alpha\gamma\beta} \neq R^{\gamma\alpha\beta}$ ...etc

Changing $^{\gamma\beta\alpha}$ by $^{\phi\Theta\psi}$ will result in the following rotation matrices for Cardan angle sequences denoted by ϕ,Θ and ψ around the axes X, Y and Z

I.XYZ decomposition

$$R\phi \cdot R\theta \cdot R\psi \rightarrow \begin{pmatrix} \cos\theta\cdot\cos\psi & -\cos\theta\cdot\sin\psi & \sin\theta \\ \sin\phi\sin\theta\cdot\cos\psi+\cos\phi\sin\psi & -\sin\phi\sin\theta\cdot\sin\psi+\cos\phi\cos\psi & -\sin\phi\cos\theta \\ -\cos\phi\sin\theta\cdot\cos\psi+\sin\phi\sin\psi & \cos\phi\sin\theta\cdot\sin\psi+\sin\phi\cos\psi & \cos\phi\cos\theta \end{pmatrix}$$

II.YXZ decomposition

$$R\theta \cdot R\phi \cdot R\psi \rightarrow \begin{pmatrix} \cos\theta\cdot\cos\psi+\sin\phi\sin\theta\cdot\sin\psi & -\cos\theta\cdot\sin\psi+\sin\phi\sin\theta\cdot\cos\psi & \cos\phi\sin\theta \\ \cos\phi\sin\psi & \cos\phi\cos\psi & -\sin\phi \\ -\sin\theta\cdot\cos\psi+\sin\phi\cos\theta\cdot\sin\psi & \sin\theta\cdot\sin\psi+\sin\phi\cos\theta\cdot\cos\psi & \cos\phi\cos\theta \end{pmatrix}$$

III.ZXY decomposition

$$R\psi \cdot R\phi \cdot R\theta \rightarrow \begin{pmatrix} \cos\theta\cdot\cos\psi-\sin\phi\sin\theta\cdot\sin\psi & -\cos\phi\sin\psi & \sin\theta\cdot\cos\psi+\sin\phi\cos\theta\cdot\sin\psi \\ \cos\theta\cdot\sin\psi+\sin\phi\sin\theta\cdot\cos\psi & \cos\phi\cos\psi & \sin\theta\cdot\sin\psi-\sin\phi\cos\theta\cdot\cos\psi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{pmatrix}$$

IV.ZYX decomposition

$$R\psi \cdot R\theta \cdot R\phi \rightarrow \begin{pmatrix} \cos\psi\cdot\cos\theta & -\sin\psi\cdot\cos\phi+\cos\psi\cdot\sin\theta\cdot\sin\phi & \sin\psi\cdot\sin\phi+\cos\psi\cdot\sin\theta\cdot\cos\phi \\ \sin\psi\cdot\cos\theta & \cos\psi\cdot\cos\phi+\sin\psi\cdot\sin\theta\cdot\sin\phi & -\cos\psi\cdot\sin\phi+\sin\psi\cdot\sin\theta\cdot\cos\phi \\ -\sin\theta & \cos\theta\cdot\sin\phi & \cos\theta\cdot\cos\phi \end{pmatrix}$$

V.XZY decomposition

$$R\phi \cdot R\psi \cdot R\theta \rightarrow \begin{pmatrix} \cos\psi\cdot\cos\theta & -\sin\psi & \cos\psi\cdot\sin\theta \\ \sin\psi\cdot\cos\phi\cos\theta+\sin\phi\sin\theta & \cos\psi\cdot\cos\phi & \sin\psi\cdot\sin\theta\cos\phi-\cos\theta\cdot\sin\phi \\ \sin\psi\cdot\sin\phi\cos\theta-\sin\theta\cdot\cos\phi & \cos\psi\cdot\sin\phi & \sin\psi\cdot\sin\theta\cdot\sin\phi+\cos\theta\cdot\cos\phi \end{pmatrix}$$

VI.YZX decomposition

$$R\theta \cdot R\psi \cdot R\phi \rightarrow \begin{pmatrix} \cos\psi\cdot\cos\theta & -\cos\phi\sin\psi\cdot\cos\theta+\sin\phi\sin\theta & \sin\phi\sin\psi\cdot\cos\theta+\cos\phi\sin\theta \\ \sin\psi & \cos\phi\cos\psi & -\sin\phi\cos\psi \\ -\cos\psi\cdot\sin\theta & \cos\phi\sin\psi\cdot\sin\theta+\sin\phi\cos\theta & -\sin\phi\sin\psi\cdot\sin\theta+\cos\phi\cos\theta \end{pmatrix}$$

???

Exercise on Cardan angle calculation

Given two matrices at time T1 and T2 (in the next slide)

- Calculate ϕ,Θ and ψ at T1 using the XYZ convention
- Calculate ϕ,Θ and ψ at T1 using the ZYX convention

- Calculate ϕ at T1 and T2 using the ZXY convention

Use the previous slide for sequence dependend rotation matrices

Rotation matrix at t1

0,098	-0,811	-0,577
-0,204	-0,584	0,786
-0,974	0,041	-0,223

Rotation matrix at t2

0,089	-0,818	-0,568
-0,2039	-0,5871	0,786
-0,967	0,066	-0,246

2.5 References

Axford, D. T., Badre, A., Johnson, J. A., & King, G. J. W. (2023). The effect of lateral collateral ligament repair tension on elbow stability: An in vitro biomechanical study. Clinical biomechanics (Bristol, Avon), 109, 106101. https://doi.org/10.1016/j.clinbiomech.2023.106101

Gava, V., Rosa, D. P., Pereira, N. D., Phadke, V., & Camargo, P. R. (2022). Ratio between 3D glenohumeral and scapulothoracic motions in individuals without shoulder pain. *Journal of electromyography and kinesiology : official journal of the International Society of Electrophysiological Kinesiology*, *62*, 102623. https://doi.org/10.1016/j.jelekin.2021.102623

Tanashi, A., Szekeres, M., MacDermid, J., & Lalone, E. A. (2022). Comparison of finger kinematics between patients with hand osteoarthritis and healthy participants with and without joint protection programs. *Journal of hand therapy : official journal of the American Society of Hand Therapists*, *35*(3), 477–487. https://doi.org/10.1016/j.jht.2020.10.010

Yildiz, T. I., Kara, D., Demirci, S., Sevinç, C., Ulusoy, B., Eraslan, L., Aksoy, T., Huri, G., & Duzgun, I. (2023). Recovery of the shoulder kinematics after reverse shoulder arthroplasty. *Clinical biomechanics (Bristol, Avon)*, *107*, 106013. https://doi.org/10.1016/j.clinbiomech.2023.106013

3 Marker based motion capture – from images to joint angles

: Motion capture - from images to joint angles

3.1 A quick introduction in motion capture:

https://www.youtube.com/watch?v=D6JMr-ZPbjQ

3.2 The basic principle of marker based photogrammetry.

3.3 Alternatives.

3.4 Applications in rehabilitation engineering.

3.5 Additional information, slides, articles

MOTION CAPTURE - FROM IMAGES TO JOINT ANGLES

Bart Jansen

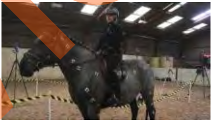
VUB VRIJE UNIVERSITEIT BRUSSEL

CONTENT

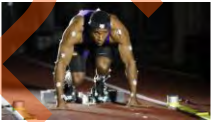
- A quick introduction in motion capture: https://www.youtube.com/watch?v=D6JMr-ZPbjQ
- The basic principle of marker based photogrammetry.
- Alternatives.
- Applications in rehabilitation engineering.

MOTION CAPTURE APPLICATIONS

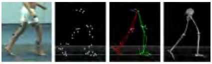
MOTION CAPTURE APPLICATIONS



MOTION CAPTURE APPLICATIONS



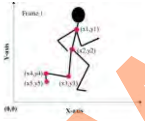
HUMAN MOTION ANALYSIS



SOME BASIC QUESTIONS

- How can we now the position of human limbs over time?
- How can we know the exact joint angles?
- How can we know the forces acting on the body?

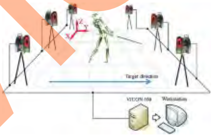
MOTION CAPTURE IN 2D



MOCAP IN 3D: TWO DIFFERENT APPROACHES

- Processing of the color image to detect the different anatomical landmarks directly.
- Using infrared markers to ease the detection.
  - Active infrared cameras with small passive markers on the body that reflect emitted light.
  - Passive cameras with active markers.
- Either way, the assumption is that from analyzing the body contours, the joint motion can be derived.

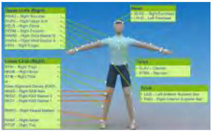
3D MOTION CAPTURE WITH INFRARED MARKERS



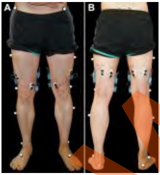
DIFFERENT STEPS TO GET JOINT ANGLES

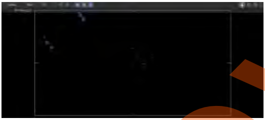
1. Place markers at anatomical landmarks.
2. Get 2D position of these markers in several camera images.
3. Get 3D position by combining the different views.
4. Define coordinate systems on different segments.
5. Get joint angles from segment coordinate systems.

STEP 1: MARKERS



STEP 1: MARKERS

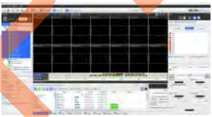




STEP 2: 2D POSITIONS OF THE MARKERS IN ALL CAMERAS

- In each image, the infrared markers are easily visible. Image segmentation is used to detect the blobs and their position in the image.
- Note that each marker is just a blob, it is not known what body part they correspond to etc.
- Correspondence finding: we need to find for every marker in the first image, where it appears in the other images.
- Markers are often missing (e.g. occlusions).

STEP 3: FROM MULTIPLE 2D TO 3D



BASIC IMAGE MODEL (GENERIC)

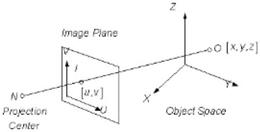


IMAGE FORMATION

DESCRIBES THE RELATION BETWEEN IMAGE AND WORLD COORDINATES

$$u = \frac{m_{11}X+m_{12}Y+m_{13}Z+m_{14}}{m_{31}X+m_{31}Y+m_{33}Z+m_{34}} \quad v = \frac{m_{21}X+m_{22}Y+m_{23}Z+m_{24}}{m_{31}X+m_{31}Y+m_{33}Z+m_{34}}$$

because

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = M \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where $M = IE$, with I the camera intrisics and E the camera extriniscs.

19

3D COORDINATES FROM MULTIPLE IMAGES

We assume that we have a point (X, Y, Z) imaged by multiple cameras i with parameters M^i and that this point results in image coordinates (u^i, v^i) for the ith image

we have: $u = \frac{m_{11}X+m_{12}Y+m_{13}Z+m_{14}}{m_{31}X+m_{31}Y+m_{33}Z+m_{34}}$ and $v = \frac{m_{21}X+m_{22}Y+m_{23}Z+m_{24}}{m_{31}X+m_{31}Y+m_{33}Z+m_{34}}$

This can be rewritten into:

$$(m_{11} - um_{31})X + (m_{12} - um_{32})Y + (m_{13} - um_{32})Z + (m_{14} - um_{34}) = 0$$
$$(m_{21} - vm_{31})X + (m_{22} - vm_{32})Y + (m_{23} - vm_{32})Z + (m_{24} - vm_{34}) = 0$$

We have two equations, with three unknowns... But we have such a pair of equations for every camera.

20

3D COORDINATES FROM MULTIPLE IMAGES

A system of $2N$ equations for N cameras:

$$\begin{bmatrix} m^1_{11} - u^1m^1_{31} & m^1_{12} - u^1m^1_{32} & m^1_{13} - u^1m^1_{32} \\ m^1_{21} - v^1m^1_{31} & m^1_{22} - v^1m^1_{32} & m^1_{23} - v^1m^1_{32} \\ m^2_{11} - u^2m^2_{31} & m^2_{12} - u^2m^2_{32} & m^2_{13} - u^2m^2_{32} \\ m^2_{21} - v^2m^2_{31} & m^2_{22} - v^2m^2_{32} & m^2_{23} - v^2m^2_{32} \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} u^1m^1_{34} - m^1_{14} \\ v^1m^1_{34} - m^1_{24} \\ u^2m^2_{34} - m^2_{14} \\ v^2m^2_{34} - m^2_{24} \\ \vdots \end{bmatrix}$$

or $Ax = b$, for which the best fit solution is $x = (A^TA)^{-1}A^Tb$

21

STEP 4: FROM MARKER POSITIONS TO JOINT ANGLES

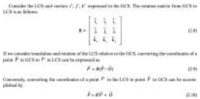

ISB recommendations on definitions of joint coordinate systems of various joints for the reporting of human joint motion—Part II: shoulder, elbow, wrist and hand

22

LOCAL AND GLOBAL COORDINATE SYSTEM

23

LOCAL AND GLOBAL COORDINATE SYSTEM


Consider the LCS unit vectors $\vec{i}, \vec{j}, \vec{k}$ expressed in the GCS. The rotation matrix from GCS to LCS is as follows:

(2.8)

If we consider translation and rotation of the LCS relative to the GCS, converting the coordinates of a point $\vec{P}$ in GCS to $\vec{P}'$ in LCS can be expressed as

$\vec{P}' = R(\vec{P} - \vec{O})$ (2.9)

Conversely, converting the coordinates of a point $\vec{P}'$ in the LCS to point $\vec{P}$ in GCS can be accomplished by

$\vec{P} = R^T\vec{P}' + \vec{O}$ (2.10)

24

ANATOMICAL REFERENCE POINTS

ISB GUIDELINES FOR THE FOREARM LCS

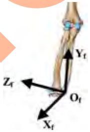
2.3.6. Forearm coordinate system—$X_f Y_f Z_f$ (see Figs. 1 and 6)

O_f: The origin coincident with US.
Y_f: The line connecting US and the midpoint between EL and EM, pointing proximally.

ISB GUIDELINES FOR THE FOREARM LCS

X_f: The line perpendicular to the plane through US, RS, and the midpoint between EL and EM, pointing forward.
Z_f: The common line perpendicular to the X_f and Y_f-axis, pointing to the right.

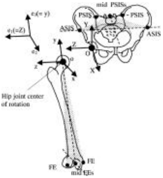
ISB GUIDELINES FOR THE FOREARM LCS



ISB GUIDELINES FOR THE HUMERUS LCS

2.3.4. Humerus (1st option) coordinate system—$X_{h1} Y_{h1} Z_{h1}$ (see 1 and 5; see also notes 1 and 2)

O_{h1}: The origin coincident with GH.
Y_{h1}: The line connecting GH and the midpoint of EL and EM, pointing to GH.
X_{h1}: The line perpendicular to the plane formed by EL, EM, and GH, pointing forward.
Z_{h1}: The common line perpendicular to the Y_{h1}- and Z_{h1}-axis, pointing to the right.



4.3. Pelvic coordinate system—XYZ (Fig. 3)

- *O*: The origin coincident with the right (or left) hip center of rotation.
- *Z*: The line parallel to a line connecting the right and left ASISs, and pointing to the right.
- *X*: The line parallel to a line lying in the plane defined by the two ASISs and the midpoint of the two PSISs, orthogonal to the *Z*-axis, and pointing anteriorly.
- *Y*: The line perpendicular to both *X* and *Z*, pointing cranially (Cappozzo et al., 1995).

31

4.4. Femoral coordinate system—xyz (Fig. 3)

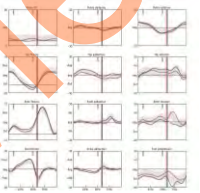
- *o*: The origin coincident with the right (or left) hip center of rotation, coincident with that of the pelvic coordinate system (O) in the neutral configuration.
- *y*: The line joining the midpoint between the medial and lateral FEs and the origin, and pointing cranially.
- *z*: The line perpendicular to the *y*-axis, lying in the plane defined by the origin and the two FEs, pointing to the right.
- *x*: The line perpendicular to both *y*- and *z*-axis, pointing anteriorly (Cappozzo et al., 1995).

32

STEP 5: JOINT ROTATION

- For every segment, we can express the rotation and translation between the GCS and the LCS according to these guidelines.
- For a given proximal and distal segment, we can compute the joint rotation by computing the rotation between the distal LCS and the proximal LCS.
- e.g. $R_{elbow} = R_{forearm} R_{humerus}^{-1}$

33



34

STRENGTHS AND WEAKNESSES

Let us make an overview now and revise it after the mocap sessions.

35

4 Markerless motion caputure

: Markerless motion capture

4.1 Introduction

4.2 How does it work

4.2.1 From rgbd to joint angles

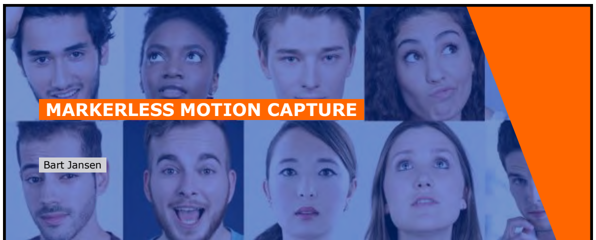
4.2.2 From 3D keypoints to joint angles

4.2.3 Model based approach

4.3 Strengths and weaknesses

4.4 Conclusions

4.5 Additional information, slides, articles



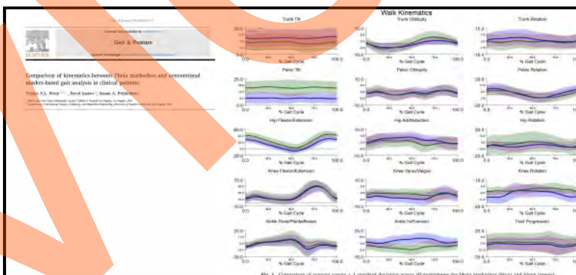
MARKERLESS MOTION CAPTURE

Bart Jansen

VRIJE UNIVERSITEIT BRUSSEL

THEIA3D

- One of Vicon's active domains of research.
- Publications in leading image and video processing conferences.
- Ring of high quality cameras, at least 6 Vue's.
- Trying to get as close as possible to marker based systems under ideal lab conditions.



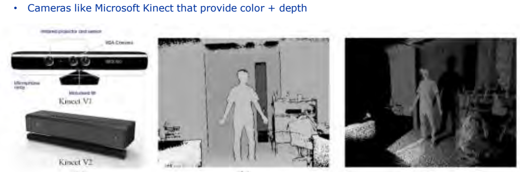
HOW DOES IT WORK?

AT LEAST FOR 1 CAMERA

FROM RGBD TO JOINT ANGLES

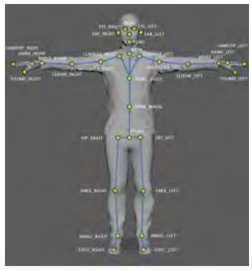
USING AN RGBD CAMERA

- Cameras like Microsoft Kinect that provide color + depth



FROM RGBD TO JOINT ANGLES
USING AN RGBD CAMERA

- Detect "keypoints" in 2D. Based on color, shape, … an AI algorithm can be trained to detect "hand", "elbow", …
- Based on the depth map, it is known how far each key point is from the camera, so a 3D position can be estimated.
- So, a limited set of 3D positions of keypoints is obtained.



FROM 3D KEYPOINTS TO JOINT ANGLES
NOT SO OBVIOUS

See the schema from yesterday.

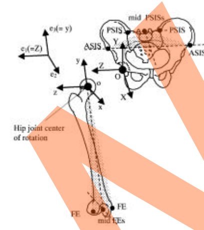
To define the femur coordinate system, we need:

- Hip joint center
- Both Epicondyles of the femur

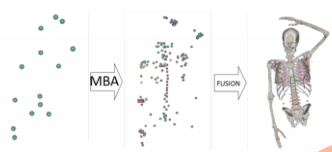
(of course other marker sets exist)

With Kinect, we have 1 Hip and 1 Knee marker

- We can define the Y axis, but not the others.
- We can assume a 1D hinge joint put knee abduction and knee rotation to 0.

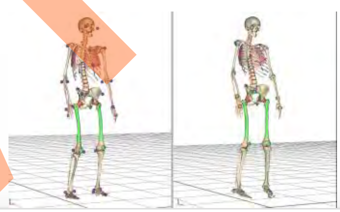


IMPROVING THE KINECT SKELETON
MODEL BASED APPROACH

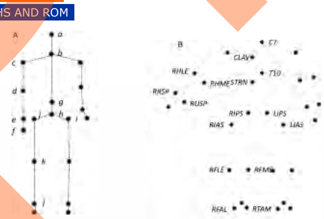


V. Sholukha et al. Journal of Biomechanics, 46(14), 2013

MORE ADVANCED BIOMECHANICAL MODELLING
KNEE SQUATING



VALIDATING KINECT
SEGMENT LENGTHS AND ROM



SEGMENT LENGTHS

Shoulder abduction, elbow flexion, hip abduction, knee flexion (N=48)

	Session 1				
	MMC	MBS	Difference	MAD	r
Arm	275 (19)	331 (19)	−55 (13)***	55 (13)	$0.76^{\dagger\dagger}$
Forearm	241 (17)	258 (17)	−16 (12)***	16 (11)	$0.72^{\dagger\dagger}$
Hand	80 (18)	84 (8)	−3 (13) $p = 0.21$	11 (8)	$0.59^{\dagger\dagger}$
Upper limb	507 (35)	566 (31)	−55 (19)***	55 (20)	$0.80^{\dagger\dagger}$
Thigh	453 (53)	408 (23)	48 (39)***	51 (35)	$0.70^{\dagger\dagger}$
Shank	368 (35)	420 (27)	−51 (27)***	53 (27)	$0.57^{\dagger}$
Foot	91 (11)	149 (11)	−57 (10)***	54 (16)	$0.44^{\dagger\dagger}$
Lower limb	825 (81)	822 (44)	4 (41) $p = 0.57$	31 (26)	$0.88^{\dagger\dagger}$

TEST-RETEST RELIABILITY SEGMENT LENGTHS

	MMC					MBS				
	ICC	SEM	RMSE	MAD	r	ICC	SEM	RMSE	MAD	r
Arm	0.89 [0.80–0.94]	5.6	8.1	6 (5)	0.72	0.84 [0.71–0.91]	7.9	11.2	9 (7)	0.91
Forearm	0.86 [0.74–0.92]	5.3	7.5	5 (5)	0.66	0.87 [0.77–0.93]	6.3	8.9	7 (6)	0.88
Hand	0.69 [0.48–0.83]	9.5	13.3	10 (8)	/	0.74 [0.55–0.85]	4.6	6.4	5 (4)	/
Upper limb	0.92 [0.84–0.96]	8.8	12.5	10 (8)	0.97	0.94 [0.89–0.97]	8.9	12.5	11 (8)	0.87
Thigh	0.95 [0.91–0.97]	11.4	16.1	12 (11)	0.76	0.89 [0.80–0.94]	9.6	13.6	17 (12)	0.89
Shank	0.70 [0.50–0.83]	16.7	23.5	16 (15)	0.77	0.91 [0.84–0.95]	8.4	11.9	9 (8)	0.91
Foot	0.21 [−0.08–0.46]	8.5	12.2	11 (7)	0.15	0.81 [0.63–0.92]	4.2	4.3	5 (5)	0.49
Lower limb	0.93 [0.87–0.96]	19.8	27.9	21 (15)	0.91	0.98 [0.96–0.99]	7.9	11.3	13 (9)	0.85
Height	0.97 [0.94 0.99]	13.4	36.2	27 (24)	0.99	0.82 [0.62 0.89]	12.5	44.1	18 (15)	0.96
Shoulder width	0.94 [0.83 0.97]	5.3	19.7	16 (11)	0.83	0.92 [0.76 0.97]	4.3	14.8	12 (9)	0.91

VRIJE UNIVERSITEIT BRUSSEL

Bonnechère, et al. Ergonomics, 57(4), 2014.

RANGE OF MOTION (ROM)

Shoulder abduction, elbow flexion, hip abduction, knee flexion

		MLS	MBS	MLS-MBS	p	LOA	R^2	CV_{ME}
Session 1								
Shoulder	ROM	111 (17)	110 (16)	0.4 (2.5)	0.301	[−5 5]	0.98	1.6
	RMSE				3 (1) [3%]			
	CMC				0.997 (0.003)			
Elbow	ROM	127 (11)	119 (8)	8 (9)	< 0.001	[−10 27]	0.33	5.3
	RMSE				11 (5) [9%]			
	CMC				0.981 (0.014)			
Hip	ROM	58 (11)	64 (10)	−5 (14)	0.021	[−34 23]	0.44	16.5
	RMSE				10 (5) [19%]			
	CMC				0.908 (0.077)			
Knee	ROM	104 (21)	111 (17)	−7 (18)	0.019	[−43 29]	0.32	11.8
	RMSE				13 (7) [12%]			
	CMC				0.956 (0.052)			

VRIJE UNIVERSITEIT BRUSSEL

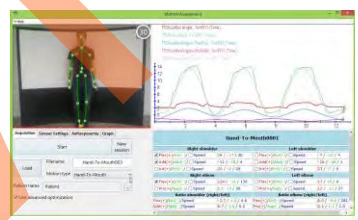
TEST-RETEST RELIABILITY ROM

		MLS				MBS			
		Session 1	Session 2	p	ICC	Session 1	Session 2	p	ICC
Shoulder	ROM	111 (17)	109 (18)	0.57	0.73	110 (16)	109 (20)	0.53	0.61
Elbow	ROM	127 (11)	126 (14)	0.69	0.70	119 (8)	119 (8)	0.79	0.77
Hip	ROM	58 (11)	60 (11)	0.283	0.84	64 (10)	64 (12)	0.878	0.83
Knee	ROM	104 (21)	97 (25)	0.078	0.66	111 (17)	108 (21)	0.115	0.89

VRIJE UNIVERSITEIT BRUSSEL

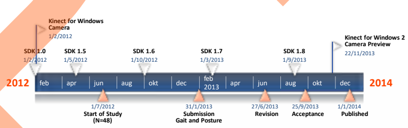
Bonnechère et al, Gait & Posture, 39(1), 2014

KINECT FOR LOW COST MOTION ANALYSIS



VRIJE UNIVERSITEIT BRUSSEL

A TIMELINE OF TECHNOLOGY VALIDATION



VRIJE UNIVERSITEIT BRUSSEL

Journal of Biomechanics 49 (2016) 2561–2565

Contents lists available at ScienceDirect

Journal of Biomechanics

journal homepage: www.elsevier.com/locate/jbiomech www.JBiomech.com

Perspective article

Cost-effective (gaming) motion and balance devices for functional assessment: Need or hype?

B. Bonnechère [a,b,c,d], B. Jansen [c,d], S. Van Sint Jan [a,b,*]

[a] Laboratory of Anatomy, Biomechanics and Organogenesis (LABO), Université Libre de Bruxelles, Brussels, Belgium
[b] Center for Functional Evaluation, Faculty of Medicine, Erasme Hospital, Université Libre de Bruxelles, Brussels, Belgium
[c] Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel, Brussels, Belgium
[d] iMinds, Department of Medical Information Technologies (MIT), Ghent, Belgium

ARTICLE INFO

Article history:
Accepted 14 July 2016

Keywords:
Functional assessment
Hardware
Methods

ABSTRACT

In the last decade, technological advances in the gaming industry have allowed the marketing of hardware for motion and balance control that is based on technological concepts similar to scientific and clinical equipment. Such hardware is attractive to researchers and clinicians for specific applications. However, some questions concerning their scientific value and the range of future potential applications have yet to be answered. This article attempts to present an objective analysis about the pros and cons of using such hardware for scientific and clinical purposes and calls for a constructive discussion based on scientific facts and practical clinical requests that are emerging from application fields.

VRIJE UNIVERSITEIT BRUSSEL

FROM RGB TO JOINT ANGLES
NO DEPTH AVAILABLE

- Detect "keypoints" in 2D. Based on color, shape, … an AI algorithm can be trained to detect "hand", "elbow", …
- But how to get 3D positions of these keypoints without depth map?
- AI systems can be trained to predict 3D position from 2D keypoints, using specific camera information. "Lifting". Not so obvious.
- But no specific hardware required. Can be webcam, smartphone, …

VUB VRIJE UNIVERSITEIT BRUSSEL 19

VUB VRIJE UNIVERSITEIT BRUSSEL

DL techniques for skeletal tracking with RGB Or DEPTH cameras

---

RGB CAMERA: 2D pose estimation

VUB VRIJE UNIVERSITEIT BRUSSEL

BOTTOM-UP APPROACH

- keypoint positions are detected on the complete image. These keypoint positions are then fitted to complete skeletal representations per person. Basically, first find the keypoints and then group them, each group will define a person.
- The bottom-up approach is much faster in detecting multiple persons within one image

[15] Cao et Al. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, XXXX

Open source project:
https://github.com/CMU-Perceptual-Computing-Lab/openpose

TOP-DOWN APPROACH

- Individual persons are initially recognized and then joint positions are detected in the corresponding image-segments. Basically, first identify each person and then estimate the pose.
- the detection of single persons is much more accurate via the top-down approach

[16] Sun et Al. Deep High-Resolution Representation Learning for Human Pose Estimation, 2019
present a novel architecture, namely HighResolution Net (HRNet), which is able to maintain high resolution representations through the whole process

Open source project:
https://github.com/leoxiaobin/ deep-high-resolution-net.pytorch

---

3D POSE ESTIMATION

VUB VRIJE UNIVERSITEIT BRUSSEL

Direct human 3D pose from a single image [13], [14],[17]:
- single input image
- fully supervised learning problem, deep architectures to directly regress the 3D coordinates of human joints from the image

Disadvantages:
- Poor generalization
- need for ground truth 3D poses to train the image

Joint 2D-3D pose estimation:
Several monocular approaches solve for 2D and 3D pose jointly.
network trained with both 2D and 3D losses

3D pose from 2D joint estimates:
First 2D pose detection and then 3D by model fitting or regression

Training with 2D-only loss [18]:
- bypass the need to annotate images with 3D ground truth labels by keeping an internal 3D representation of the pose but training based on 2D reprojection losses.
- Better generalization as they do not rely on 3D annotated images that can only be captured in studios;

Multi-view human pose [19]:
Multi-viewtrackers combine data from different camera views to estimate the temporal evolution of objects across a monitored area. Data to be combined can be represented by object features (such as position, color and silhouette) or by object trajectories in each view.

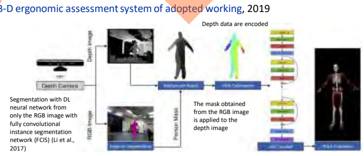
VUB VRIJE UNIVERSITEIT BRUSSEL

DEPTH CAMERA: Joint Angles

VUB VRIJE UNIVERSITEIT BRUSSEL

[13] Abobakr et Al., RGB-D ergonomic assessment system of adopted working, 2019

- Deep convolutional neural network to analyze the posture and predict body joint angles from a single depth camera (depth and RGB image). The temporal information is not needed, it's a single frame.
- Generalized DL model thanks to the training with synthetic images. The corresponding ground truth joint angles have been generated using inverse kinematics modeling stage with OpenSim ☺
- Validated in real environment obtaining a joint angle mean absolute error (MAE) of 3.19 ± 1.5°
- The objective is to calculate the RULA score based on single image



Synthetic depth generation
- Images generated via rendering animated human models created with MakeHuman software.
- Model "animated" (single frame) according to VICON data. From Carnegie Mellon University (CMU) mocap database are obtained 3650 postures.
- Rendering of the models using 8 virtual Kinect depth sensors. The scene is created and rendered using BlenSor

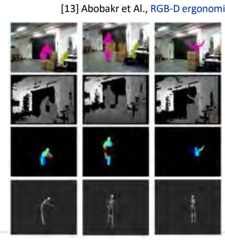
Encoding of depth data
- Data normalization.
- the depth values are transformed to (0 255 –) range and a jet color map is applied.

VUB VRIJE UNIVERSITEIT BRUSSEL

DEPTH CAMERA: Joint angles

VUB VRIJE UNIVERSITEIT BRUSSEL

[13] Abobakr et Al., RGB-D ergonomic assessment system of adopted working, 2019



The forth row shows predicted joint angles applied to the biomechanical model in OpenSim

Better than Kinect:
Kinect doesn't work with occlusion, it has problem even with self occlusion

VUB VRIJE UNIVERSITEIT BRUSSEL

DEPTH CAMERA: tracking for surgery

[14] Liu et Al., Automatic Markerless Registration and Tracking of the Bone for Computer-Assisted Orthopaedic Surgery, xxxx

- Automatic, markerless method for registration and tracking of the femur based on depth images and DL
- Depth camera acquires RGB and Depth images.
- NN trained to first localise the surgical target using the RGB image, then segment the target area of the corresponding depth image, from which the surface geometry of the target bone can be extracted.
- The extracted surface is then compared to a pre-operative model of the same bone for registration (ICP algorithm between the individuated points and the model*)
- The tracking is possible by repeating all the steps (5-6 Hz, must be increased)
- accuracy measurements against an optically tracked ground truth resulted in a mean translational error of 2.74 mm and a mean rotational error of 6.66°. In the experimental testing the knee moves

NN TRAINING

*By using optical marker and a probe it's possible to create a model, label the acquired images and train the NNs (ROI localization network for RGB and depth image segmentation network)

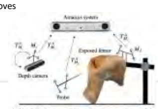


FIGURE 1: Experimental setup for image collection of the knee anatomy.

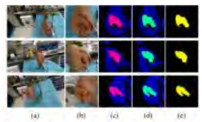


FIGURE 4: Examples of localisation and segmentation results of RGB and depth images. Three rows represent three groups of results from different viewing positions. (a): ROI localisation in RGB images. The centres of the red boxes are the predicted ROI positions, and the size of the red boxes (128 × 128) is used to crop the depth images. (b): RGB images corresponding to the cropped depth images. (c): Depth images (blue) with labelled pixels (ground truths, magenta). (d): Depth images (blue) with predicted femur pixels (cyan). (e): The ground truths (red), the predictions (green) and their overlays (yellow).

DEPTH CAMERA: Joints tracking

[17] Büker et Al, HRDepthNet: Depth Image-Based Marker-Less Tracking of Body Joints, 2021

keypoints detection relies on evaluating RGB data, while depth information is only considered for the estimation of the depth of the joints—representing the state-of-the-art

- HRDepthNet to detect keypoints of persons in depth images instead of RGB data. The model is based on the HRNet CNN model [16], which is retrained for annotated depth images.
- To evaluate the sensitivity of using an HRNet-like model for the keypoint detection via depth images instead of RGB images, the algorithm's sensitivity is evaluated using COCO's evaluation metrics and Is compared to the sensitivity of HRNet rendered on the RGB images of the same dataset.
- Experimental test with one person walking (max distance: 10 m)

Preprocessing & Data preparation

- To train the model the depth image must be segmented. Background subtraction gray value normalization, image cropping, and scaling are conducted.
- ground truth: Keypoint positions were manually annotated on the depth images with the VGG Image Annotator (VIA) software using the corresponding RGB images. They acquired RGBD images to train the model.

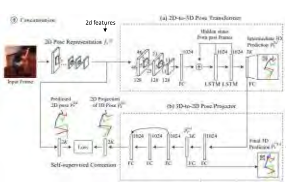
Figure 1. Color-coded background elimination of a point cloud. The elimination in the several axes are coded as follows: x-axis green, y-axis (floor) light blue, z-axis purple.

RGB CAMERA: 3D pose estimation

[18] Wang et Al., 3D Human Pose Machines with Self-supervised Learning, 2019
(suggested by Lubos as state of the art)

Limitations of the State of the art approaches for pose estimation with depth camera:
- Time consuming network
- Not enough data (depth images) to train
- Some works uses pre knowledge (constraints)

The architecture combines:
- CNN (spatial info)
- RNN (temporal info)
- Self-supervised correction
- Training with 2D labelled images dataset MPII Human Pose



- taking each RGB frame as input, the model first predicts the 2D poses.
- The 2D-to-3D pose transformer module transform the learned pose representations from the 2D domain to the 3D domain. regression of the intermediate 3D poses via two stacked LSTM. The module is designed to estimate intermediate 3D poses for each frame by incorporating temporal dependency in the image sequence.
- 3D-to-2D pose projector module. the 2D projections of 3D poses and the predicted 2D poses should be identical, the minimization of their dissimilarities is the learning objective.
- extensive evaluations on two publicly available benchmarks: Human3.6M and HumanEva-I.

RGB CAMERA: 3D pose estimation

[18] Wang et Al., 3D Human Pose Machines with Self-supervised Learning, 2019

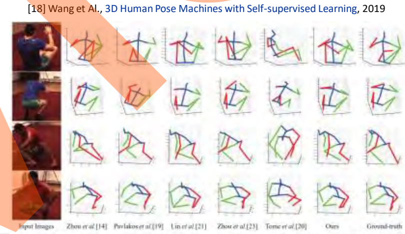


Fig. 4: Qualitative comparisons on the Human3.6M dataset. The 3D poses are visualized from the side view, and the cameras are depicted. The results from Zhou et al. [14], Pavlakos et al. [19], Lin et al. [21], Zhou et al. [23], Tome et al. [20], our model and the ground truth are illustrated from left to right. Our model achieves considerably more accurate estimations than all the compared methods. Best viewed in color. Red and green indicate left and right, respectively.

MULTIPE RGB CAMERAS: 3D pose estimation

[19] Tome et Al., 20.Rethinking Pose in 3D Multi-stage Refinement and Recovery for Markerless Motion Capture, 2018

- Huber loss based robust estimator for fusing multi-view 2D pose predictions into a coherent 3D pose
- Unlike existing 3D frameworks, this is not simply done at the end of a pipeline for 2D joint estimation, but is iterated through multiple-stages
- Often, monocular approaches use synthetic dataset or available labelled data (i.e. Human3.6M). To help address this issue, we demonstrate how unlabelled data can be labelled by our algorithm and augment the datasets used for the training of existing methods, leading to overall better performance on standard bench

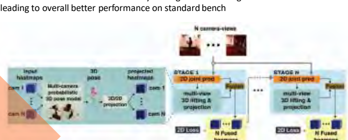


Figure 1. Multi-stage architecture of our proposed multi-camera 3D human pose network. Each stage takes images from all the camera views and the set of per-image 2D joints (expressed as heatmaps) predicted in the previous stage and outputs a refined prediction. In each stage, the 2D predictions from all views are used to reconstruct a single 3D pose, consistent with all camera views. This 3D pose is projected back into the image and used to improve predictions in the next stage. See section 3.1 for more details.

LITEPOSE

- Openpose and many other 2D keypoint detectors are super heavy
- Litepose works on mobile phone and augmented reality glasses.

RELEVANCE
SEE FRIDAY!

The use of commercial video games in rehabilitation: a systematic review

Bruno Bonnechère[a,b,c], Bart Jansen[b,c], Lubos Omelina[b,c,d] and Serge Van Sint Jan[a]

The aim of this paper was to investigate the effect of commercial video games (VGs) in physical rehabilitation of motor functions. Several databases were screened (Medline, SAGE Journals Online, and ScienceDirect) using combinations of the following free-text terms: commercial games, video games, exergames, serious gaming, rehabilitation games, PlayStation, Nintendo, Wii, Wii Fit, Xbox, and Kinect. The search was limited to peer-reviewed English journals. The beginning of the search time frame was not restricted and the end of the search time frame was 31 December 2015. Only randomized controlled trial, cohort, and observational studies evaluating the effect of VGs on physical rehabilitation were included in the review. A total of 4728 abstracts were screened, 275 were fully reviewed, and 126 papers were eventually included. The following information was extracted from the selected studies: device type, number and type of patients, intervention, and main outcomes. The integration of VGs into physical rehabilitation has been tested for various pathological conditions, including stroke, cerebral palsy, Parkinson's disease, balance training, weight loss, and aging. There was large variability in the protocols used (e.g. number of sessions, intervention duration, outcome measures, and sample size). The results of this review show that in most cases, the introduction of VG training in physical rehabilitation offered similar results as conventional therapy. Therefore, VGs could be added as an adjunct treatment in rehabilitation for various pathologies to stimulate patient motivation. VGs could also be used at home to maintain rehabilitation benefits. *International Journal of Rehabilitation Research* 00:000–000

International Journal of Rehabilitation Research 2016, 00:000–000

Keywords: exercises, motivation, new technology, rehabilitation

[a]Laboratory of Anatomy, Biomechanics and Organogenesis (LABO), Université Libre de Bruxelles, [b]Department of Electronics and Informatics – ETRO, Vrije Universiteit Brussel, Brussels, [c]Department of Future Health, iMinds, Ghent, Belgium and [d]Institute of Computer Science and Mathematics, Slovak University of Technology, Bratislava, Slovakia

Correspondence to Bruno Bonnechère, MSc, Laboratory of Anatomy, Biomechanics and Organogenesis (LABO), Université Libre de Bruxelles, Lennik Street 808, 1070 Brussels, Belgium
e-mail: bbonnech@ulb.ac.be

Received 21 April 2016 Accepted 6 July 2016

GAMES FOR HEALTH JOURNAL: Research, Development, and Clinical Applications
Volume 9, Number 5, 2020
© Mary Ann Liebert, Inc.
DOI: 10.1089/g4h.2019.0097

Original Article

The Effect of a Rehabilitation Specific Gaming Software Platform to Achieve Individual Physiotherapy Goals in Children with Severe Spastic Cerebral Palsy: A Randomized Crossover Trial

Sophie Decavele, PT, MSc,[1] Els Ortibus, MD, PhD,[1,2] Anja Van Campenhout, MD, PhD,[1–3] Guy Molenaers, MD, PhD,[1–3] Bart Jansen, PhD,[4,5] Lubos Omelina, PhD,[4,5] and Inge Franki, PT, PhD[1,2]

Abstract

Cerebral palsy (CP) is the most common cause of permanent neurological disabilities in children. Many children require long-term daily physiotherapy (PT), and videogaming is a promising tool to increase motivation in rehabilitation. The short- and medium-term effects of an intervention with rehabilitation specific videogames were evaluated on individually defined therapy goals, gross motor function, and motivation. Thirty-two children with bilateral spastic CP, Gross Motor Function Classification level III–IV, and 6–15 years were randomized into an intervention group (regular PT and gaming) or a control group (regular PT), followed by a crossover. The effects of both training periods (each 12 weeks) were compared using the Goal Attainment Scale (GAS), Trunk Control Measurement Scale (TCMS), Pediatric Balance Scale (PBS), Gross Motor Function Measure-88 (GMFM-88), and Dimensions of Mastery Motivation Questionnaire (DMQ). After 3 months follow-up, children were retested using the GMFM, TCMS, and PBS. The GAS change scores were significantly higher after the intervention compared to the control period (8.5 and 2.4, $P < 0.001$). The change scores for standing exercises (3.85 and 0.22, $P = 0.04$) and dynamic sitting balance (5.9 and −1.7, $P < 0.001$) were also significantly higher. After 3 months follow-up the results did not persist. A combined approach of regular PT and rehabilitation specific gaming showed significant effects on individually defined therapy goals, dynamic sitting balance, and standing exercises. However, the lack of persistent effect indicates that continuous individual goal-oriented PT with the addition of gaming is needed.

LET US TRY THE KINECT
STRENGHTS AND WEAKNESSES

?

CONCLUSIONS

- Vicon has put very promising (clinically relevant) markerless motion capture on the market.
  - But most of the disadvantages of their marker based approach still remain.
- There is a decade of Kincet like solutions on the market.
  - Super cheap, easy to use, but limited quality data.
  - Better suited for rehab exercises than for assessment.
- There is active research on smartphone based solution.
  - Measure whenever, wherever by whoever. E.g Partner films person walking after total knee replacement.
  - Very promising, but even lower data quality.

5 Dynamic CT for MSK applications

: Dynamic CT for MSK applications

5.1 Introduction

5.2 Basic principles in Medical imaging

5.3 Dynamic CT for MSK applications

5.3.1.1 Cadaver experiments

5.3.1.2 Study

5.3.1.2.1 Scan protocol optimization

5.3.1.2.2 Image analysis

5.3.1.2.3 (Pre) clinical studies

5.4 Questions and answers

5.5 Additional information, slides, articles