Geometric & Graphics Programming Lab: Lecture 8

Alberto Paoluzzi

October 29, 2016

1. Workshop N.3
2. Jupyter notebook required
3. Minimal git/github instructions

Workshop N.3

Parametric concrete stairs

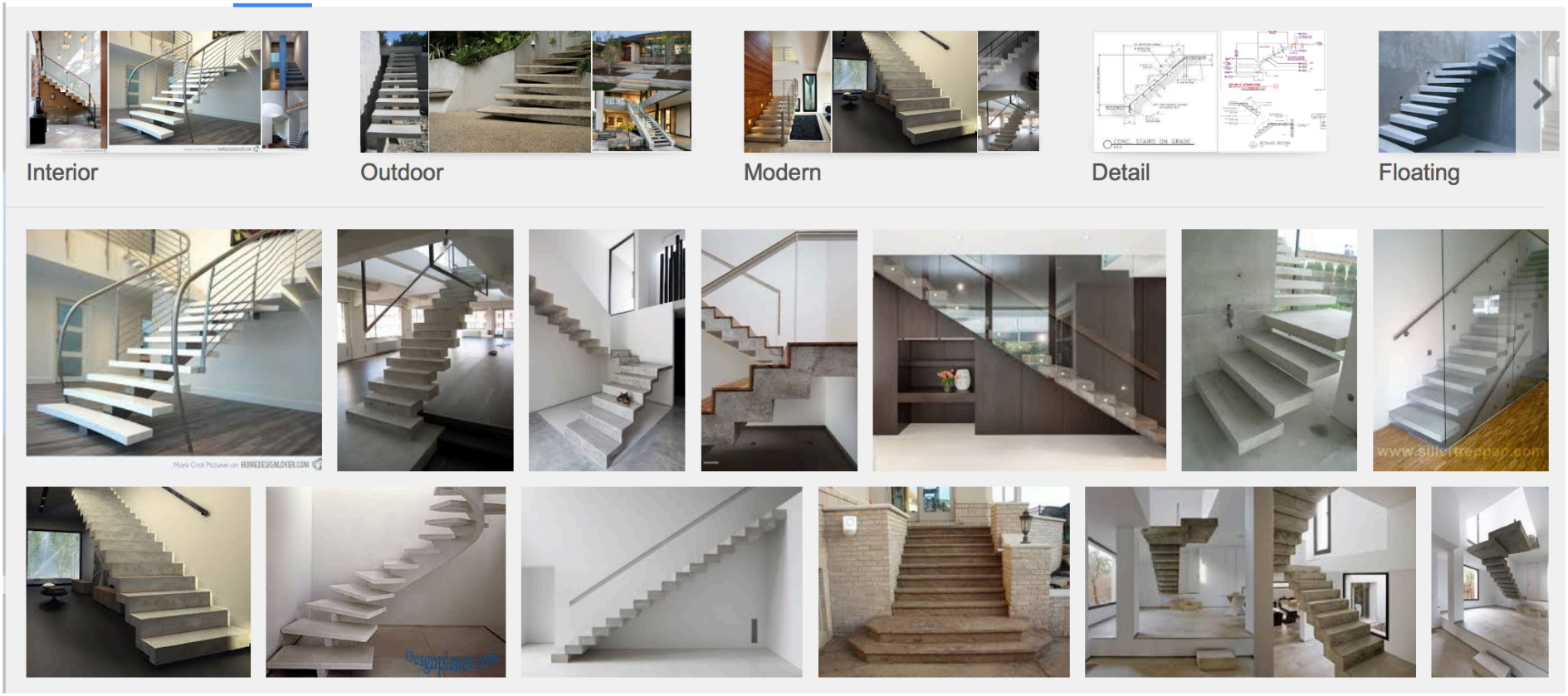

Figure 1: Images from Google

Look at some examples

- Double stairs design
- Building stairs with landings
- Floating concrete stairs
- Stair Design From The Ground Up.
- Concrete stairs design

Stair terminology (1/2)

- Wikipedia
- Types of stairs
  - Straight Stairs.
  - Straight Stair with central landing.
  - L Shaped Stair.
  - L Shaped Winder Stairs.
  - Spiral Stairs.
  - Curved Staircase.

Stair terminology (2/2)

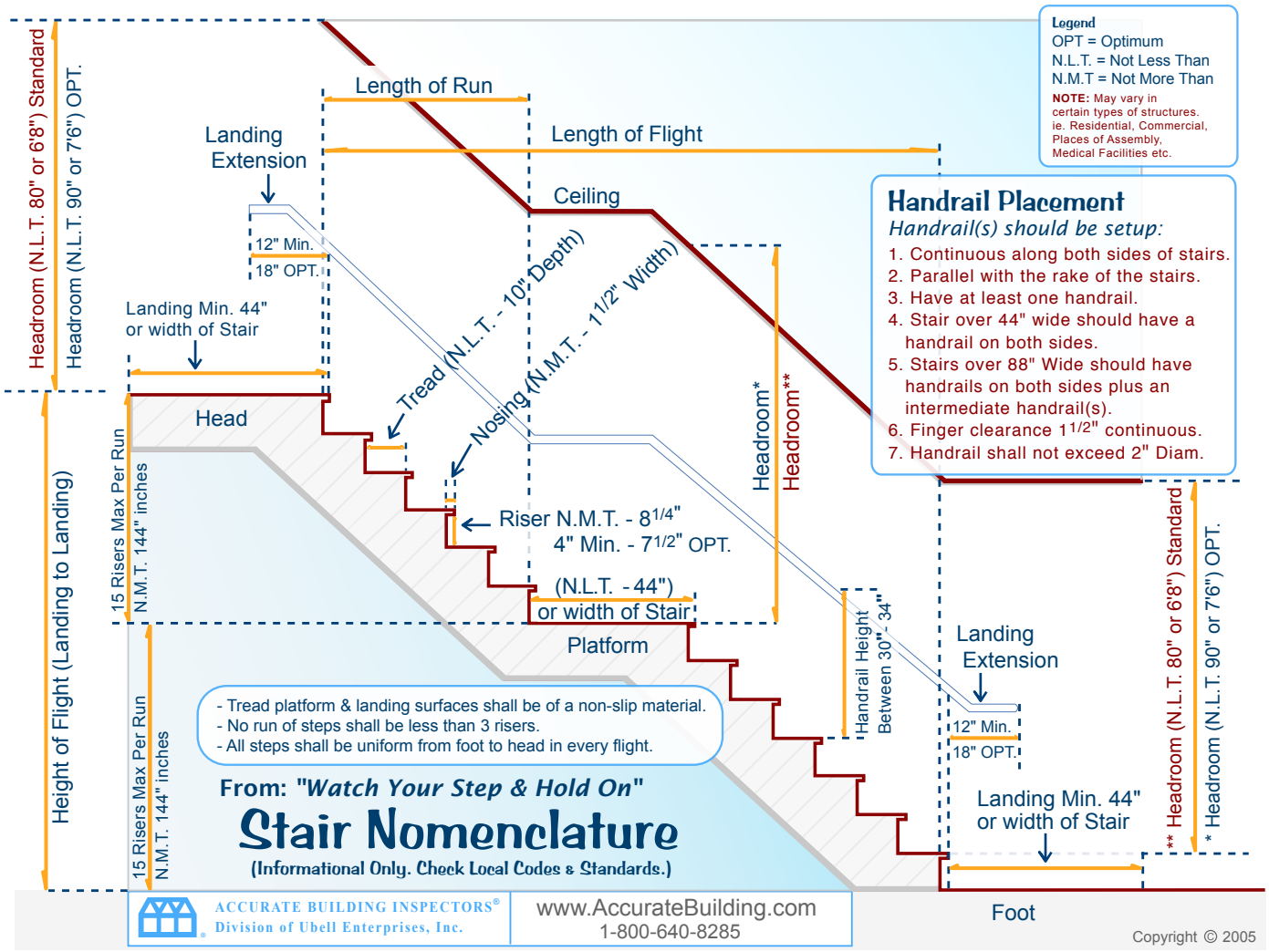

Figure 2: stair nomenclature

Step dimensioning (Blondel formula) (1/2)

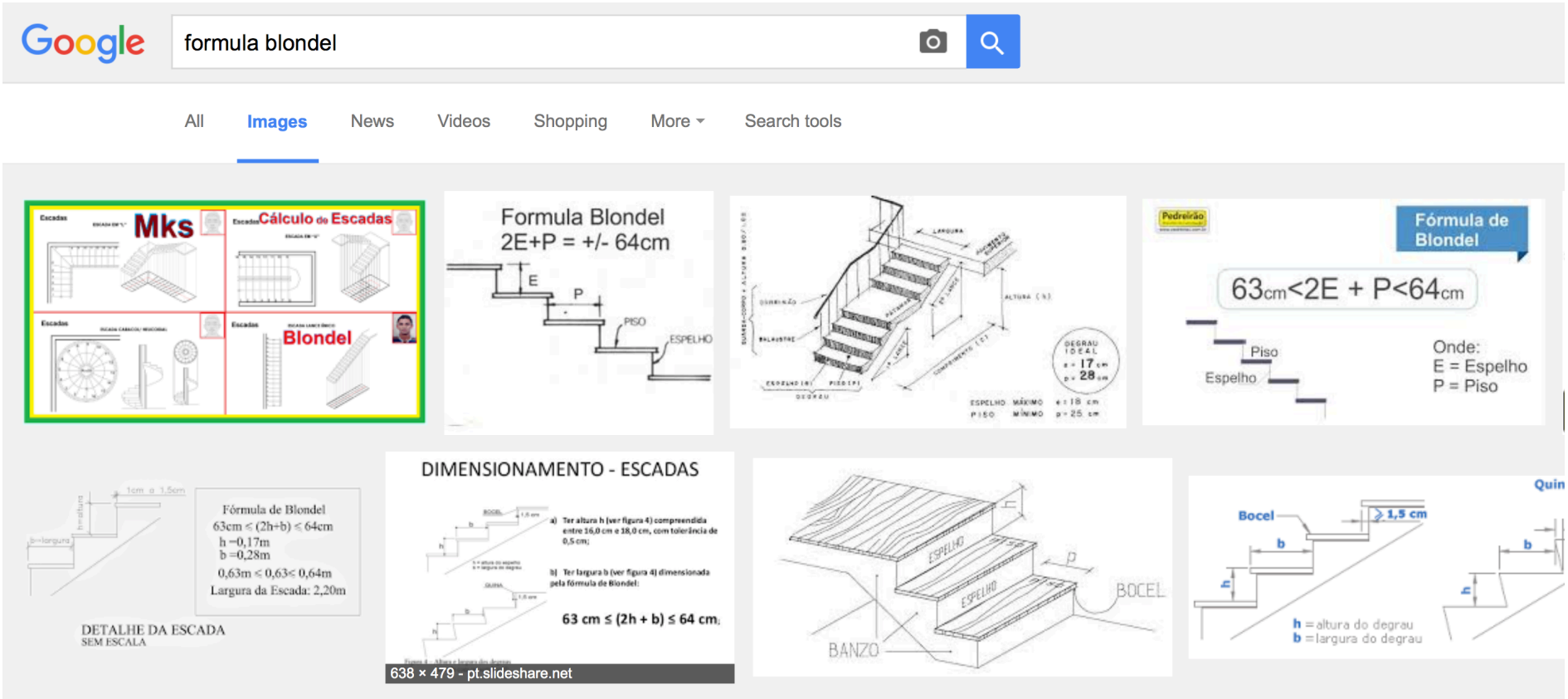

Figure 3: Step dimensioning (Blondel formula)

Step dimensioning (Blondel formula) (2/2)

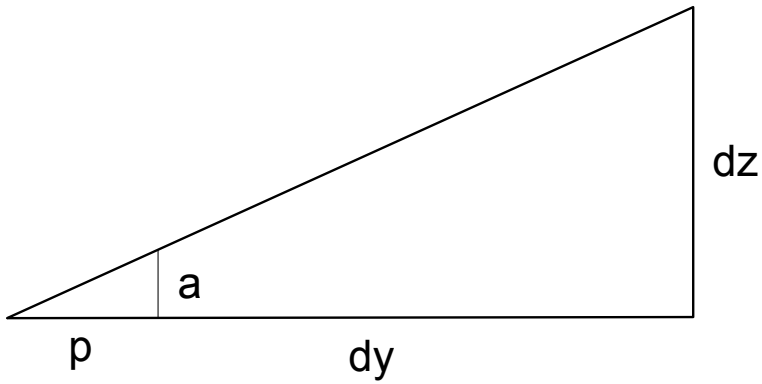

Figure 4: Relationship of similarity between stair rise (dz) and run (dy), and step (riser height (a), tread depth (p))

1. Nicolas-François Blondel in the last volume of his "Cours d'architecture" (1675–1683)

$$2a + p = 63 \quad (1)$$

$$a/p = dz/dy \quad (2)$$

2. number n of steps:

$$n = \text{round}(dz/a) \quad (3)$$

3. step dimensions:

$$riser = dz/n \quad (4)$$

$$tread = dy/n \quad (5)$$

Requirements

- Write a single notebook, named `workshop_03.ipynb`
- Choose a notebook Title, for example `<my_stair_type>`
- Start the notebook with a web reference and one/more image/s of your type of stairs (i.e. your chosen kind of stair model)
- List the variables used in your code, with a textual definition
- Provide a short description of the geometric method you are going to implement
- Include the coding of a single Python function named `ggpl_<my_stair_type>`
- Provide only 3 formal parameters, named `dx,dy,dz`, respectively
- Provide the images generated by two or more executions with different actual parameters.
- Use measures in meters (m)

Hints to solution

1. First look for formulas linking slope to riser to treads according to good practice
2. Then generate the 2D model of each single (parametric) trait (using `MKPOL`, `POLYLINE`, `JOIN`, `CUBOID`, etc.)
3. Then extrude the 2D models to generate the 3D parts (using `Q`, `QUOTE`, `PROD`, `INSR(PROD)`, etc.)
4. Then assemble the parts (using `STRUCT`, `T`, `R`, `S`, `MAT`, and Boolean ops if only strictly necessary)
5. Finally compute the containment box (`BOX([1,2,3])(hpc_obj)`, `SIZE([1,2,3])(hpc_obj)`, `SKELETON`, etc), compare with the actual parameters, and possibly apply a global scaling to compensate . . .

Style specs (1/2)

- produce a notebook file, of type `.ipynb` (The ipynb file extension is associated with the IPython notebook and/or Jupiter, a rich architecture for interactive computing written in Python and available for various platforms.)

Style specs (1/2)

- output: a single HPC value
- use meaningfull identificators (variables and parameters)
- use camelCase ids
- add Python `docstrings` (google for it)
- produce a single notebook file, named workshop_03.ipynb
- file path: `your_repo/2016-10-28/workshop_03.ipynb`

Jupyter notebook required

Notebook tutorial

Notebook Basics

Minimal git/github instructions

Minimal git/github instructions (1/2)

create your local repository

```
$ mkdir development
$ cd development
$ git clone https://github.com/your-account/ggpl
$ cd ggpl
$ mkdir 2016-10-28
$ cd 2016-10-28
$ touch workshop_03.ipynb
```

Minimal git/github instructions (2/2)

commit your work

```
$ git add -A .
$ git commit -m "add a short note to commit"
$ git push origin master
```