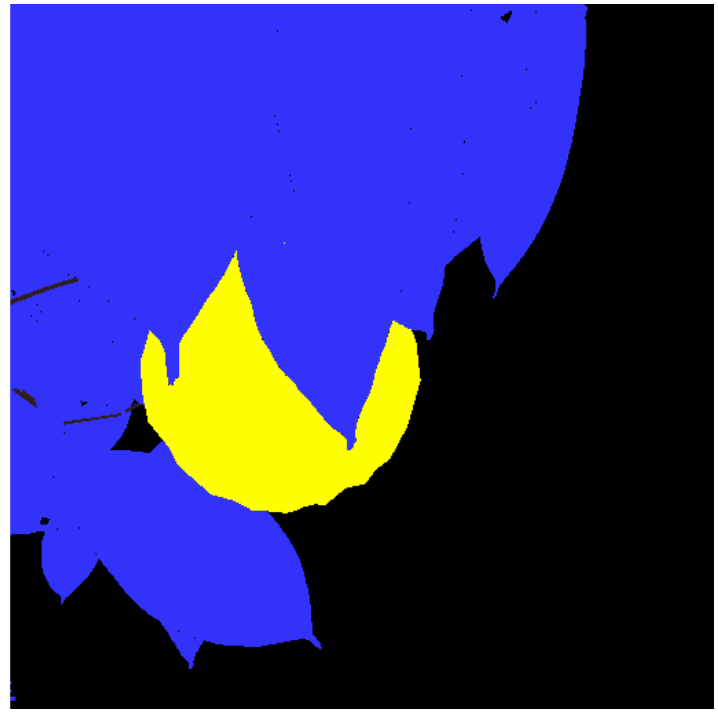
Toward a Procedural Fruit Tree Rendering Framework for Image Analysis



Thomas Duboudin, Maxime Petit, Sridhar Ragupathi, Liming Chen

ÉCOLE CENTRALE LYON

LIRIS

IAMPS

Learn-Real

Toward a Procedural Fruit Tree Rendering Framework for Image Analysis

Typical applications of AI/Deep learning for agriculture

- Crop supervision :
  - growth and counting
  - infestation or disease detection
- Robotic crop manipulation :
  - fruits harvesting
  - crop watering and pruning

Typical applications of AI/Deep learning for agriculture

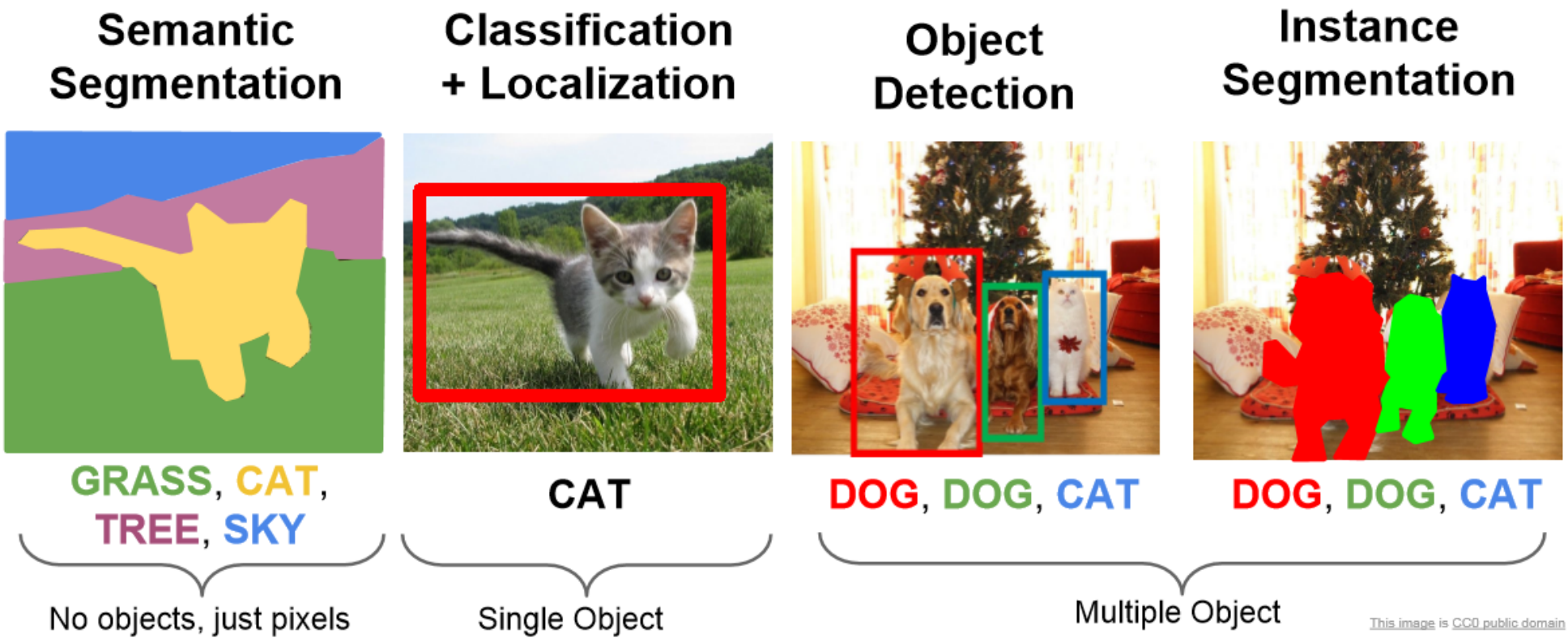
- Crop supervision :
  - growth and counting
  - infestation or disease detection
- Robotic crop manipulation :
  - fruits harvesting
  - crop watering and pruning

$\Longrightarrow$ **Related computer vision tasks :**

object detection, semantic segmentation, depth estimation

Deep Learning

- Neural Networks based algorithms are State-of-the-Art in several of the main computer vision related tasks

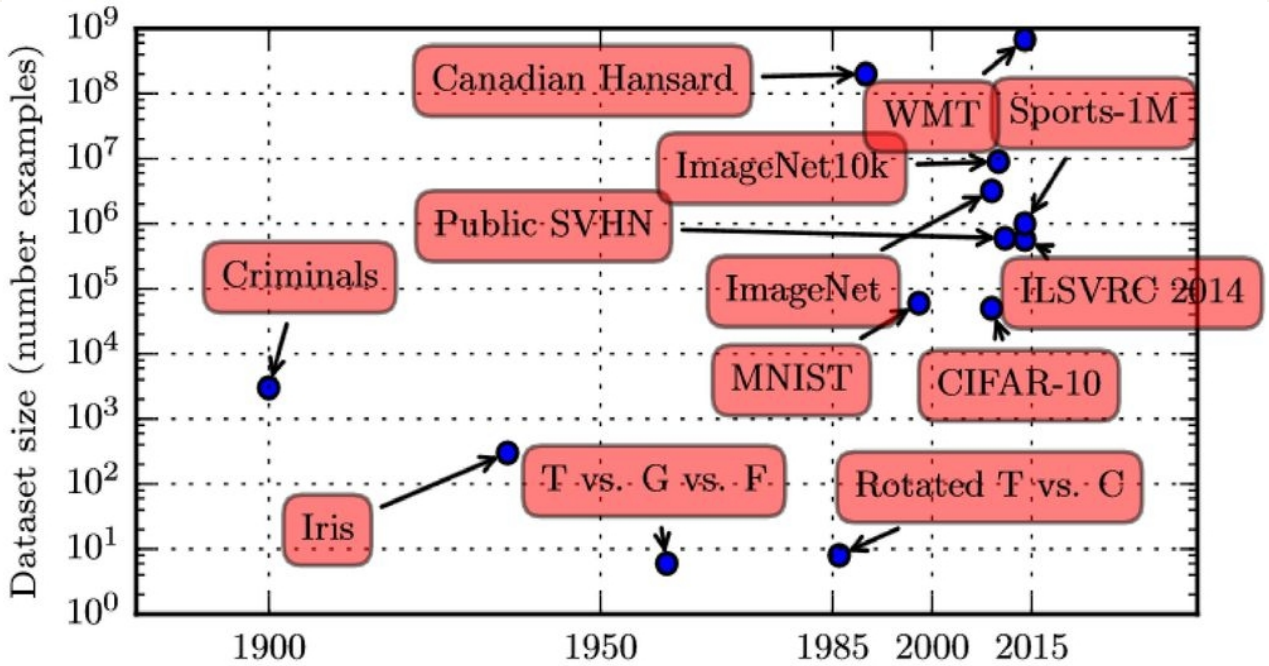


Fei-Fei Li & Justin Johnson & Serena Yeung - May 2017

Deep Learning

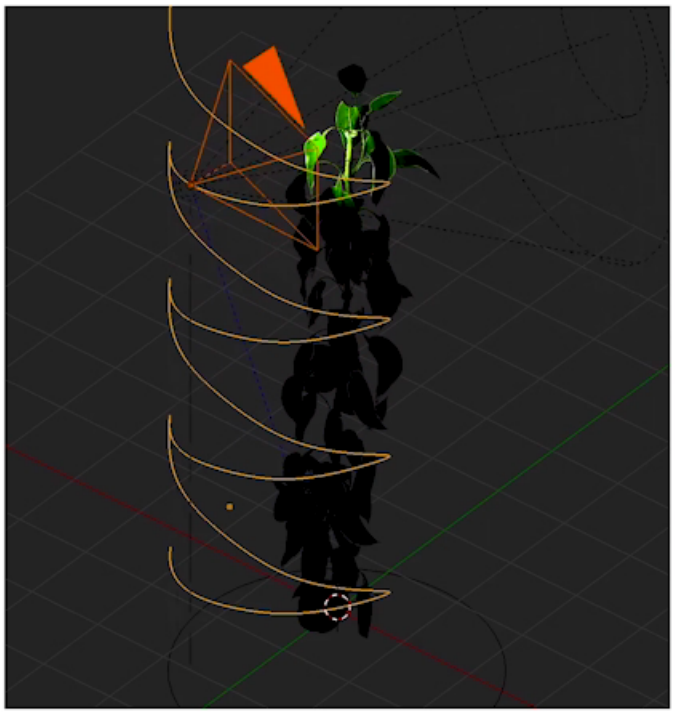
- 2 factors responsible for this success :
  - **Large amount of available annotated data.**
  - High computing capabilities (GPU)
- How to proceed when not enough relevant data is available ?

$\Longrightarrow$ **Use synthetic data instead !**



Our problem : robotics for fruits harvesting

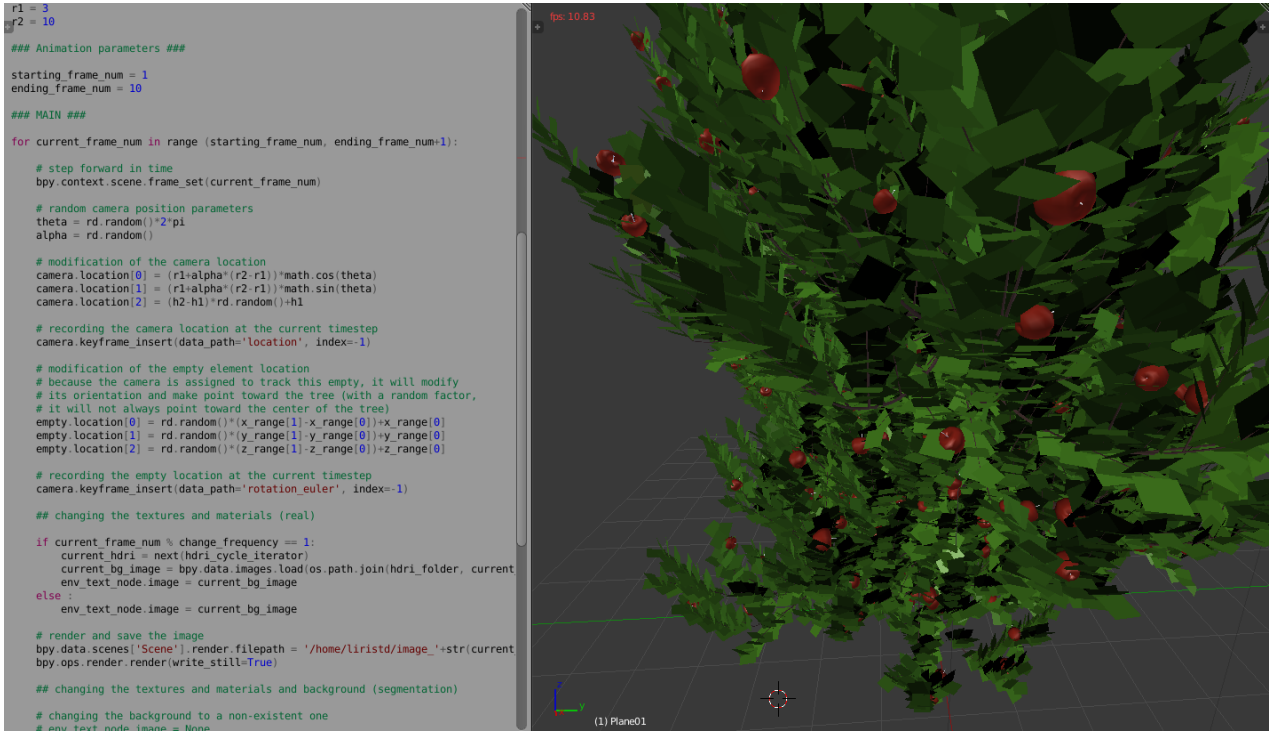
- Close to the tasks of semantic segmentation and depth estimation
- Data available :
  - 1 synthetic dataset for Bell Peppers only (Sweeper Project)
  - No large-scale annotated real dataset (< 1000 images)
  - Lots of non-annotated data



Practical fruit tree rendering framework

- Fast rendering of annotated images (perfect unbiased ground truth)
- Diverse images generation (variations of pov, background, *etc.*)
- High reusability (*e.g.* easy to change the leaves and fruits type)

$\Longrightarrow$ **Allows personalized large-scale dataset generation**



```
r1 = 3
r2 = 10

### Animation parameters ###

starting_frame_num = 1
ending_frame_num = 10

### MAIN ###

for current_frame_num in range (starting_frame_num, ending_frame_num+1):

    # step forward in time
    bpy.context.scene.frame_set(current_frame_num)

    # random camera position parameters
    theta = rd.random()*2*pi
    alpha = rd.random()

    # modification of the camera location
    camera.location[0] = (r1+alpha*(r2-r1))*math.cos(theta)
    camera.location[1] = (r1+alpha*(r2-r1))*math.sin(theta)
    camera.location[2] = (h2-h1)*rd.random()+h1

    # recording the camera location at the current timestep
    camera.keyframe_insert(data_path='location', index=-1)

    # modification of the empty element location
    # because the camera is assigned to track this empty, it will modify
    # its orientation and make point toward the tree (with a random factor,
    # it will not always point toward the center of the tree)
    empty.location[0] = rd.random()*(x_range[1]-x_range[0])+x_range[0]
    empty.location[1] = rd.random()*(y_range[1]-y_range[0])+y_range[0]
    empty.location[2] = rd.random()*(z_range[1]-z_range[0])+z_range[0]

    # recording the empty location at the current timestep
    camera.keyframe_insert(data_path='rotation_euler', index=-1)

    ## changing the textures and materials (real)

    if current_frame_num % change_frequency == 1:
        current_hdri = next(hdri_cycle_iterator)
        current_bg_image = bpy.data.images.load(os.path.join(hdri_folder, current
        env_text_node.image = current_bg_image
    else :
        env_text_node.image = current_bg_image

    # render and save the image
    bpy.data.scenes['Scene'].render.filepath = '/home/liristd/image_'+str(current
    bpy.ops.render.render(write_still=True)

    ## changing the textures and materials and background (segmentation)

    # changing the background to a non-existent one
    # env_text_node.image = None
```

Methods

1. Scene generation of parametrized fruit tree [1]
2. Image acquisition from in-engine camera trajectory

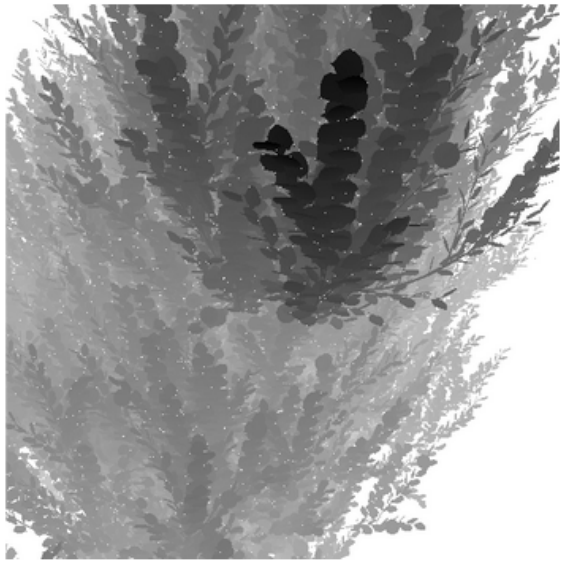
Scene parameters	**Rendering parameters**
Tree architecture	Camera point of view
Type of fruits	Background image
Sizes and shapes of fruits	Lightning conditions
Sizes and shapes of leaves	Rendering quality
Fruits and leaves textures	
Position and number of fruits	
Position and number of leaves	
Type of labels wanted	

[1] Based on the Sapling Tree Gen Blender Addon by Andrew Hale & Aaron Buchler

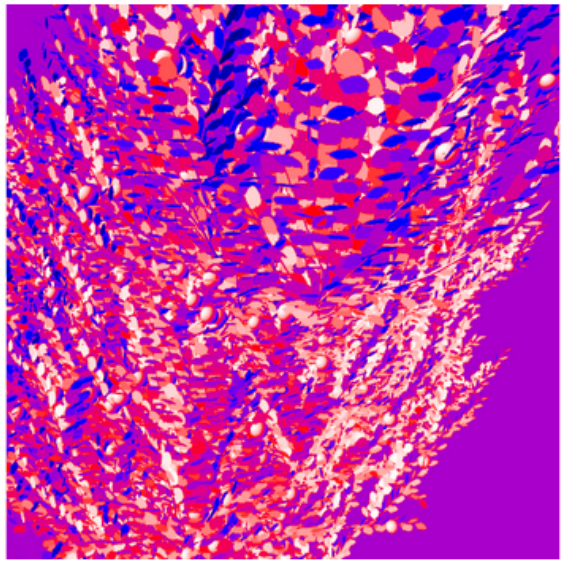
Methods

- "Low" computational requirements : NVIDIA GTX 1080
- < 30s for one image + its annotation (512 x 512 pixels)
- Several kinds of annotations : semantic map, depth map, *etc.*
- Constrained camera trajectory is possible (mimicking an embedded camera on a robot)

Rendered Image



Depth Map



Normal Map

Results : random selection

Results : real and fake comparison

Results : variations

day/night

background

fruit type

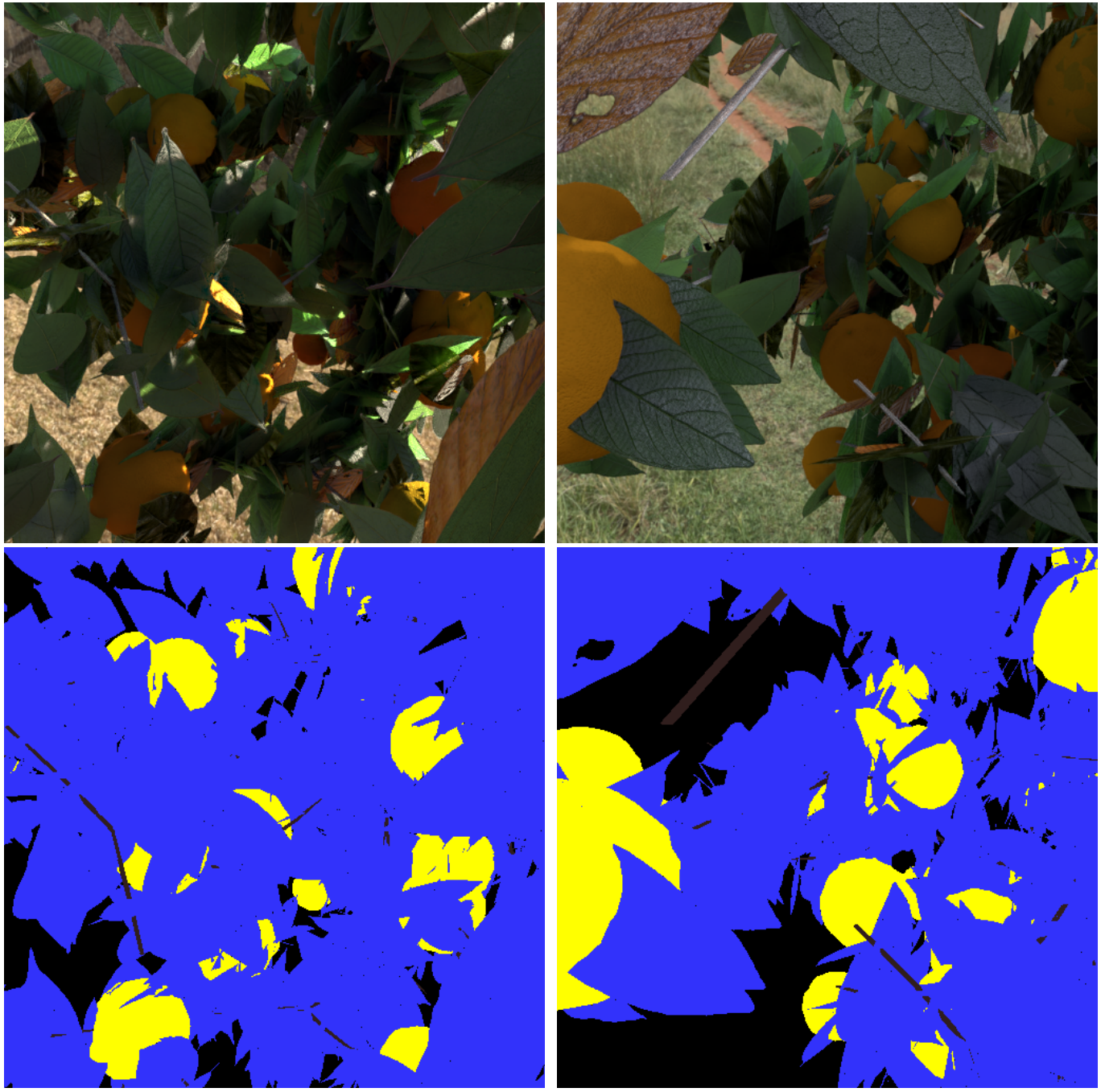
intra-scene variation








Results : semantic segmentation labels



Conclusion

- Source code available here :

  https://github.com/tduboudi/IAMPS2019-Procedural-Fruit-Tree-Rendering-Framework

- Can be used to generate around 10k images in one day for your personalized computer vision task related to fruits harvesting

Future Works

- Use of **Domain Adaptation** methods
- Add physical and biological properties (maturation, disease, *etc.*)
- Improving the simulator reusability (feedbacks welcome)

Acknowledgements

Thanks for your attention !

This work is supported by the french National Research Agency (ANR), through the ARES labcom (grant ANR 16-LCV2-0012- 01) and by the CHIST-ERA EU project "Learn-Real".

Learn-Real

$\Longrightarrow$ https://learn-real.github.io/