DART
Automatic image ground truth labeling

Pau Cano Ribé

Document Changes

1.1	28/10/2020	Added Table of Contents, figure numeration, Document Changes Table. Small format changes
1.2	23/11/2020	Small changes to the database expansion section and to the scene processing section to make them easier to understand.

Table of Contents

1. Introduction

Image labeling is possible in DART, though it is complicated to set up, and requires the creation of a new material database. To avoid both of these things, this guide presents the automated scripts necessary to do so, and comes with a custom material database for labeling, including six colors. To add more colors, refer section 4 of the guide, *Extending labels*.

2. Scene material

Each material of your simulation will be converted to a single color in this process. This means that any objects sharing the same material will share the same label.

Take for example the scene of figures 2.1 and 2.2:

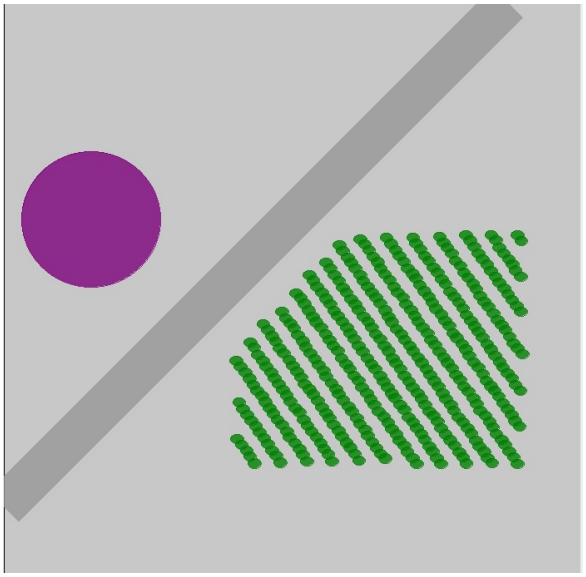

Fig 2.1: Scene overview as seen in the parameters editor.

Fig 2.2: Materials used in the scene, as seen in the parameters editor

In this scene we have a field of vines, in green, a road, in grey, and an olive tree, in purple. We have four materials: Leaf, Bark, Ground, and Road. Both the olive tree and the vines share the same materials, both use the Leaf material for leaves, and the Bark material for bark and trunk.

As such, both the olive tree and the vines would have the same label, as they share the same materials. If you want to have a label for the vines, and another for the olive tree, for example, you would need to create two new materials, one for the leaves of the olive tree, and one for its trunk. For information on how to create and assign materials, refer to *Simple Field Scene Creation* guide, section 4, *Material Creation*.

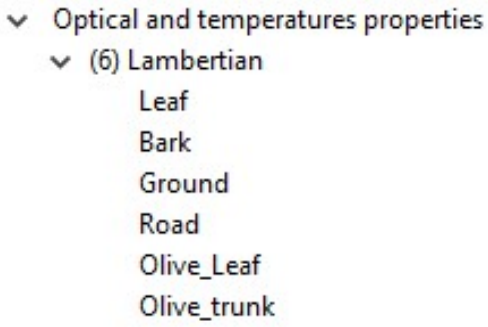

Fig 2.3: Materials used in the scene, including the two new materials, as seen in the parameters editor.

In any case, as they are different types of vegetation, they should always have different materials, as a good practice.

3. Scene Processing

The first step in the process is copying the simulation folder and assigning it a proper name, such as *SimulationName_GT*. It can be either copied in the file explorer as any other file or folder would be, or you can open the simulation in DART, click the *Simulation* menu, and click *Save as*, and saving it as *SimulationName_GT.* This can be skipped, but it would mean losing the materials' settings for the objects, as well as sun direction.

Once the simulation folder has been created, create a text file containing the material correspondence, inputMaterials.txt. This text file will only contain a list of colors in order, each corresponding to the material in dart in the same index. First row contains the color of the first material, second row contains the color of the second material, etc.

For each line of the text file, the input value can be the name of the color, either capitalized or not, or the ID of the color. The valid entries for each of the lines are:

Red	red	0
Green	green	1
Blue	blue	2
Yellow	yellow	3
Purple	purple	4
Cyan	cyan	5

Any other value will be treated as "Red". If there are not enough lines in the file to assign each material, the color will default to red.

For example, if we want to label the ground as green, the vines as red, the road as blue, and the olive tree as purple, the file for the simulation's six materials' could be:

```
Red
red
Green
2
Purple
4
```

Once this file is created and filled, the script just needs to be called. Its general call is:

```
python simulation_segmentation.py -s SimulationFolderPath -o OutputImage.png -i
inputMaterials.txt
```

And, as an example, it could be:

```
python simulation_segmentation.py –s
C:\Users\user\DART\user_data\simulations\testSimulation –o
C:\Users\user\DART\user_data\simulations\testSimulation\output.png -i
C:\Users\user\DART\user_data\simulations\testSimulation\materials.txt
```

The script will show what is working and when it has finished each of its phases, as well as errors if the parameters had not been passed correctly.

When the process has finished, the labeled and processed image will be in the output path indicated previously.

4. Extending labels

In the case that having 6 labels is not enough, it is possible to add more colors to the database and processing scripts. This can be done in four phases:

- Color calculation
- Database extension
- Python script color list extension
- Python script if-else section extension

4.1. Color calculation

Given any color, it's conversion to reflectance, which can be used inside DART, is easy and quick. Each of the RGB bands must be divided by 2.55, resulting in a value between 0 and 100.

Take for example, Orange. In RGB, orange can be represented by (255, 128, 0). In a scale of 0 to 100, this would be (100, 50, 0). 100% red wavelength reflectance, 50% green wavelength reflectance and 0% blue wavelength reflectance

4.2 .Database extension

The current database, which can be opened and modified easily using SQLite DB Browser, has 7 tables, one for each of the color labels, and one for internal SQLite data.

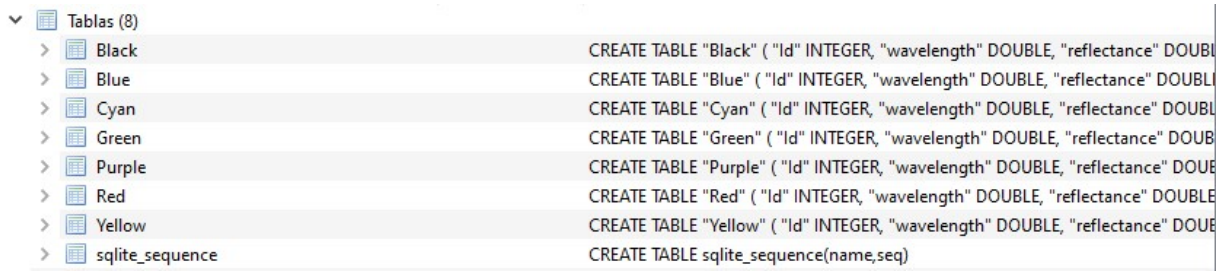

Fig. 4.1: Table structure of the database *lambertian_label.db*

Each of the color labels tables includes 1544 rows, for reflectance and transmittance values of wavelengths between 349nm and 14µm. The values corresponding to each of the main colors are:

- Blue: values between 0.4347 and 0.4572
- Green: values between 0.5415 and 0.5662
- Red: values between 0.6402 and 0.6788

You can add another table, in this case, for color orange, by executing the SQL command:

```
CREATE TABLE "Orange" (
	"Id"	INTEGER,
	"wavelength"	DOUBLE,
	"reflectance"	DOUBLE,
	"direct_transmittance"	DOUBLE,
	"diffuse_transmittance"	DOUBLE,
	PRIMARY KEY("Id" AUTOINCREMENT)
)
INSERT INTO Orange SELECT * FROM Black;
```

This creates and empty table for the color orange, and copies all the contents of the table corresponding to color black, to make sure that all the rows necessary for DART to interpret this table correctly already exist.

Once this is done, the reflectance values need to be adjusted. In this case, for orange, it should be:

- Values between 0.4347 and 0.4572: 0
- Values between 0.5415 and 0.5662: 50
- Values between 0.6402 and 0.6788: 100

4.3. Python color list extension

At the beginning part of labelled_process.py, a list of lists is defined, called “colors”:

```
colors=[[255,0,0],
	[0,255,0],
	[0,0,255],
	[255,255,0],
	[255,0,255],
	[0,255,255]
	]
```

Extending this list only requires adding another list item to it, preceded by a comma. The item added should be the RGB representation of the color. In this case, to add orange to the script, the list would be:

```
colors=[[255,0,0],
	[0,255,0],
	[0,0,255],
	[255,255,0],
	[255,0,255],
	[0,255,255],
	[255,128,0]
	]
```

4.4. Python if-else extension

The last necessary step to add another label is to modify the if-else section of the file material_segmentation.py, the script that sets up the dart materials. Between lines 48 and 60 is an if-else section, which reads the contents of a text file and sets the material accordingly, such as:

```
elif fileValue == "Cyan" or fileValue == "cyan" or fileValue == "5":
        material.set("ModelName", "Cyan")
```

To add another color to the script, just copy one of these lines paste it before the last else expression

```
else:
        material.set("ModelName", "Red")
```

and modify it to fit the desired color. In this case, to add orange, taking into account the name of the table created before, it should be:

```
elif fileValue == "Cyan" or fileValue == "cyan" or fileValue == "5":
        material.set("ModelName", "Cyan")
elif fileValue == "Orange" or fileValue == "orange" or fileValue == "6":
        material.set("ModelName", "Orange")
else:
        material.set("ModelName", "Red")
```

After all of this is done, the new color should function without problems, and should be used the same as any other color, just writing it in the text file.

5. Requirements

The file “Lambertian_label.db” that comes with this guide must be under the folder C:\Users\user\DART\database, or under the database folder of your DART installation, along with the other DART databases, like “Lambertian_mineral.db” or “Lambertian_vegetation.db”

Python must be installed, with matplotlib and skimage installed.