Manual

# HTPheno

A Novel Image Analysis Pipeline for
High-Throughput Plant Phenotyping

July 14, 2010

# Contents

# Chapter 1

# Basic step-by-step setup

To enable a versatile applicability of **HTPheno** to different high-throughput phenotyping setups, we decided to follow a modular approach: several configuration files allow to adjust the plugin to the user's needs. All configuration files are located in the same directory as the **HTPheno**-plugin (ImageJ/plugins/HTPheno/).

Before going into the details of the configuration files, have a look at the simplified processing workflow:

1. a pixel with colour values ⇒ is classified into a class c
2. a class c ⇒ is represented by a natural number n
3. a number n ⇒ is mapped to a colour in the classified image

One advantage of this scheme is that different classes, e.g. for dark and light plant parts, can be mapped to the same output colour.

Please read the following step-by-step guide thoroughly to set up **all** the needed configuration files for the side view images and the top view images, respectively (see also tutorial *Setup; http://htpheno.ipk-gatersleben.de/tutorials/setup.htm*).

## 1.1 Define the image object classes

At first decide which classes of objects the image contains (in side view images and top view images, respectively).

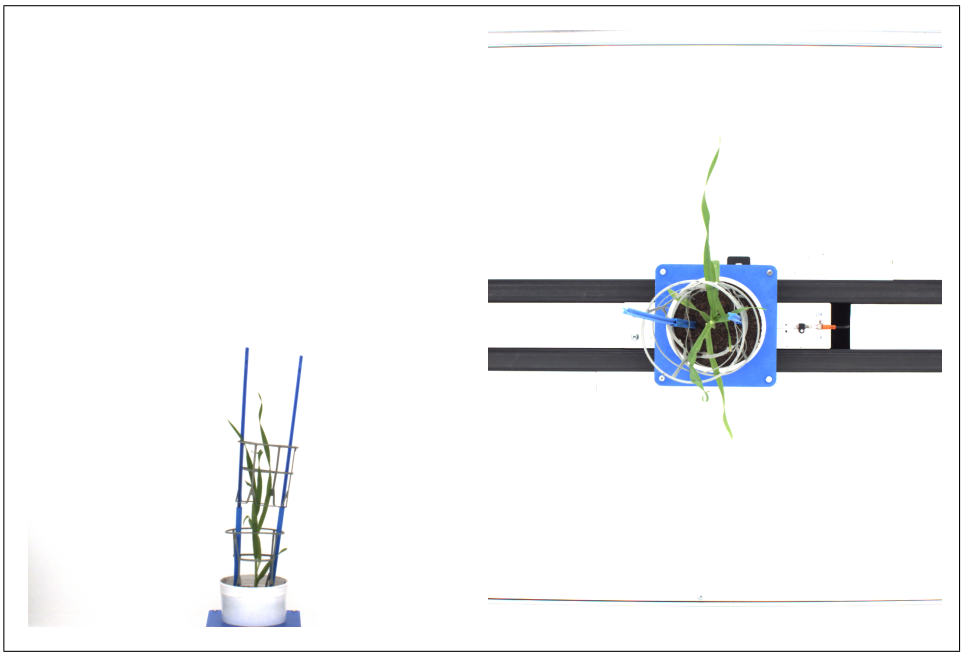

**Figure 1.1:** Example images side view image (left) and top view image (right).

Insert the object classes into the configuration files for example with ImageJ (see tutorial *Setup; http://htpheno.ipk-gatersleben.de/tutorials/setup.htm*). For the following example configuration consider these classes (names arbitrarily chosen):

- side view image: plant-light, plant-dark, cages, sticks, carrier, background;
- top view image: plant-light, plant-dark, cages, sticks, soil, background, conveyor belt, carrier;

The configuration files would look as follows (define one class per line):

```
plant-light
plant-dark
cages
sticks
carrier
background
```

**Listing 1.1:** "config_color2class_side.csv"

```
plant-light
plant-dark
cages
sticks
carrier
soil
background
conveyor belt
carrier
```

**Listing 1.2:** "config_color2class_top.csv"

At this stage no colour ranges for theses classes are necessary. They will be created later with the help of the **HTPcalib** function (see Section 2) for side view images and top view images, respectively.

### With added colour ranges

After calibrating the classes with **HTPcalib** the first file would for example look like this:

```
plant-light,71,152,88,164,49,122,34,76,45,151,88,164
plant-dark,27,95,35,106,20,79,39,76,54,156,35,106
cages,119,189,122,186,128,193,0,251,0,38,128,193
sticks,27,145,53,180,102,255,151,172,81,206,102,255
background,255,255,255,255,255,255,0,0,0,0,255,255
```

**Listing 1.3:** "config_color2class_side.csv with colour ranges"

**Parameters in detail**

The order of the parameters is as follows:

1. class name

   RGB colour values:

2. red-channel mininum, red-channel maximum
3. green-channel mininum, green-channel maximum
4. blue-channel mininum, blue-channel maximum

   HSV colour values:

5. hue mininum, hue maximum
6. saturation mininum, saturation maximum
7. value mininum, value maximum

For a pixel to be classified as class *plant-light* it has to comply to the following conditions for its RGB and HSV colour values:

$$71 \leq R \leq 152 \quad 88 \leq G \leq 164 \quad 49 \leq B \leq 122$$
$$34 \leq H \leq 76 \quad 45 \leq S \leq 151 \quad 88 \leq V \leq 164$$

**Notice:** As dealing with 8bit images here, each value ranges between 0 and 255. The HSV values are normalised.

**Notice:** Instead of defining *plant-light* and *plant-dark* as separate classes, it is possible to define only one class *plant*. While performing HTPcalib the user has the alternative to define several sets of different colour ranges to the class *plant* (see tutorial *HTPcalib; http://htpheno.ipk-gatersleben.de-/tutorials/HTPcalib.htm*). Please remember to adjust the other configuration files accordingly.

## 1.2 Define the regions

To segment the images into regions where objects can be expected, define regions in the *config_regions_side.csv* and *config_regions_top.csv* files, respectively.

A region is specified by its dimensions and by the objects that are expected to be found in. It is also possible to choose between a rectangular and an oval shape of the region, and to specify a colour for its outline as hexadecimal RGB-colour value. A tool to choose colours from a colour pallet, displaying the appropriate hexadecimal colour code is for example the online tool "The HTML Color Wheel Coordinator" (see *http://www.html-color.net/*).

An example configuration file for a top view image:

```
1  background-region,R,0,1233,0,1623,FFFFFF,background
2  plant-region,R,0,1233,0,1623,FFFFFF,plant-light,plant-dark
3  carrier-region,R,450,790,635,973,ff8040,carrier
4  beltLT-region,R,0,455,675,744,00ffff,conveyorbelt
5  beltLB-region,R,0,455,862,928,00ffff,conveyorbelt
6  beltRT-region,R,785,1233,679,744,00ffff,conveyorbelt
7  beltRB-region,R,785,1233,866,932,00ffff,conveyorbelt
8  soil-region,O,480,757,670,932,ff0000,soil
9  sticks-region,R,352,896,544,1054,000000,sticks
10 cages-region,R,400,840,580,1020,ff00ff,cages
```

**Listing 1.4:** "config_regions_top.csv"

### Parameters in detail

The order of the parameters for a region entry is as follows:

- the region name (arbitrarily chosen)
- the type of the region ("R"ectangular or "O"val)
- x-start, x-end
- y-start, y-end
- the colour of the regions outline

- a list of allowed objects in that region (named exactly as in *config_color2class_*.csv* files)

The *central-region* is of type "R"ectangular and contains all pixel that lie within the rectangle specified by 450 to 780 x-coordinates and 850 to 1580 y-coordinates in the side view image. Dimensions can be measured directly with ImageJ (see tutorial *Setup; http://htpheno.ipk-gatersleben.de/-tutorials/setup.htm*). Pixel of the object classes *sticks* or *cages* are only classified within that rectangle.

**Notice:** Regions are processed in the order of their appearance in the configuration file. Use this advantageously and carefully select an appropriate order.

**Notice:** These two configuration files are the only ones that are not necessarily needed to conduct the classification (could be left empty). **HTPheno** will then use the *config_color2class_*.csv* files as a fallback.

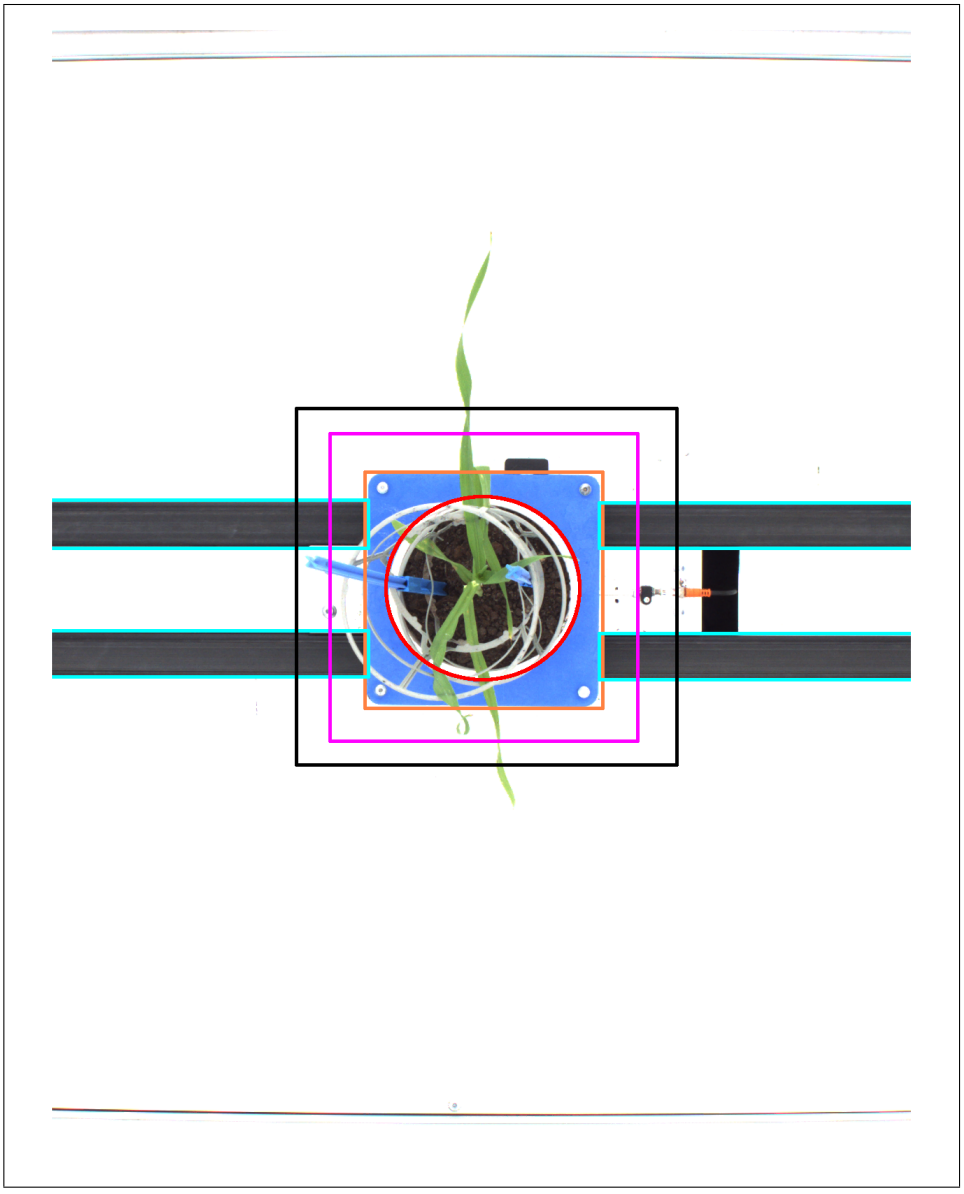

**Figure 1.2:** A more complex example of regions visualised.

## 1.3 Define the internal mapping

Different classes (e.g. *plant-dark* and *plant-light*) can belong to one object (*plant*). Therefore define the internal mapping in the *config_class2number.csv* configuration file.
Within that file assign an integer number to each object which has been defined in the *config_color2class_*.csv* files (see tutorial *Setup; http://HTPheno.-ipk-gatersleben.de/tutorials/setup.htm*).

For our previous example the file would look like this:

```
1 plant-light,1
2 plant-dark,1
3 sticks,2
4 cages,3
5 background,4
6 carrier,5
7 conveyor belt,6
```

**Listing 1.5:** "config_class2number.csv"

**Notice:** This single file contains the information for side view images and top view images.

## 1.4 Define the output colour mapping

Up to now all the information required for a successful classification is collected. The only information still missing is how to colourise each of the classified image parts.
The colour mapping definition directly connects to the previous section:
For each NUMBER representing one (or more classes) in the *config_class2-number.csv* configuration file define a hexadecimal RGB-colour value in the *config_number2color.csv* configuration file.

```
1,00ff00
2,00ffff
3,ff00ff
4,ffff00
5,00ffff
6,000000
```

**Listing 1.6:** "config_number2color.csv"

The *plant-dark* and *plant-light* classes, corresponding to number **1**, will be green (00ff00), the *conveyor belt*, corresponding to number **6**, will be painted black (000000), etc.

**Notice:** The classification workflow is now complete: colour-to-class, class-to-number, and number-back-to-output-colour.

## 1.5 Define the scaling

The scaling parameters are necessary to convert the measured values for the phenotypic parameters from pixel to millimetre. Two parameters are necessary:

- the number of pixel,
- and the corresponding number of millimetre.

To obtain these values measure an image part with ImageJ (e.g. measure width of the plant carrier with ImageJ in pixel (256 pixel), see tutorial *Scaling; http://htpheno.ipk-gatersleben.de/tutorials/scaling.htm*), and then also measure the same part in reality in millimetre (195 mm).
If these scaling parameters are determined for side view images and top view images, insert them into the *config_scaling_side.csv* and *config_scaling_top.csv* configuration files, respectively.

Those two values, pixel first, then a colon, then the millimetre, form the scaling calibration files.

```
1 256:195
```

**Listing 1.7:** "config_scaling_side.csv"

Here the top view 332 pixel correspond to 195 millimetre:

```
1 332:195
```

**Listing 1.8:** "config_scaling_top.csv"

After successfully finishing the setup of all the necessary configuration files, start the **HTPcalib** function to calibrate the colour ranges for each class defined in the *cpnfig_class2color_*.csv* configuration files.

# Chapter 2

# Using HTPcalib

A good colour range calibration is essential for reliable image processing results. Therefore sufficient time should be spent on this step.

The already created very basic *config_color2class_side.csv* and *config_color2-class_top.csv* configuration files in the ImageJ/plugins/HTPheno directory look like the following example:

```
1 plant-light
2 plant-dark
3 cages
4 sticks
5 background
```

**Listing 2.1:** "config_color2class_side.csv"

The function **HTPcalib** will now assist you in determining the colour ranges for each class defined in the above configuration files. Here as an example the workflow for side view image calibration is shown (see tutorial *HTPcalib; http://htpheno.ipk-gatersleben.de/tutorials/HTPcalib.htm* for top view calibration):

- start ImageJ
- start the **HTPcalib** function (Plugins → HTPheno → HTPcalib)
- select one or more side view images (if the image name contains "side" HTPcalib allocates the colour ranges to the configuration file *config_color2class_side.csv*)

- For each of the defined classes **HTPcalib** allows to select several pixel of that class in the image to determine its colour range.
- After each click **HTPcalib** visualises the current colour range by marking all matching pixel in green.
- Once finished with the colour range selection of a class store the results by pressing ENTER. **HTPcalib** asks then to whether create another colour range set for this class to further refine it. Otherwise **HTPcalib** proceeds with the next defined class.
- It is possible to take back several steps by pressing ESC, in case you accidentally clicked on the wrong pixel in that class or too much other pixel are assigned.
- To see temporarily the original image (without markups) press and hold the TAB key.

**Notice:** Repeat the above steps for the top view images.

### Hints for a good calibration

- Do not cover too different image parts of the same class with only one colour range calibration as it might then also cover other parts that lie in between the defined colour ranges (see Fig. 2.1). In this case add another colour range set for the class by pressing ENTER. A better way is rather to create two classes (as already suggested, e.g. *plant-dark* and *plant-light*) and calibrate them separately. As another possibility define only one class *plant* and then create several different colour range sets for it with **HTPcalib** (see tutorial *HTPcalib; http://htpheno.ipk-gatersleben.de/tutorials/HTPcalib.htm*).

- When calibrating a class, try to keep the already selected image areas contiguous before extending them further.

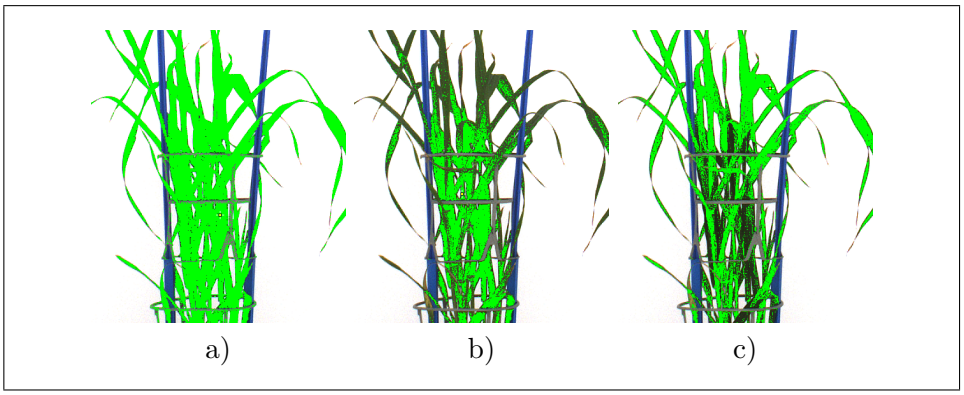


**Figure 2.1:** Examples of using **HTPcalib**: a) A too wide colour range of dark and light plant parts also selects cage-pixel. b) and c) With two classes, one for light and dark plant parts, respectively, usually a more exact calibration can be achieved.

# Chapter 3

# Using HTPheno

After completing the and the calibration, start **HTPheno** in the ImageJ Plugins menu (Plugins → HTPheno → HTPheno (single file) or HTPheno (complete directory)) and choose either the "single file" or the "complete directory" mode. Select an image or a directory in the open dialog box.

**Notice:** Only PNG files that contain either "side" or "top" in their file names can be processed.

**HTPheno** allows to adjust the scaling calibration in case the scaling parameters differs from the configuration files (*config_scaling_*.csv*).

Now **HTPheno** processes the image(s) automatically. For each selected file **HTPheno**

- opens and displays the image,
- marks the regions according to the *config_regions_*.csv* definitions,
- classifies it according to the *config_color2class_*.csv* and *config_regions_*.csv* definitions,
- further improves the achieved classification,
- calculates and displays the image parameters,
- builts up an image stack consisting of 6 images:
  1. original image,
  2. regions where the objects of the image are expected to be found in,

3. object segmentation by colour classification,
4. object extraction,
5. morphology which lowers the noise level by performing an opening,
6. analysis results displaying the obtained phenotypic parameters of the plants,

- shows the result parameters in a result table,
- process the next image, if applicable.

**Notice:** The results (image stack and result table) are stored in the same directory as the original image(s). The results file will be named *HTPheno_analysis_results_date–time.csv* (e.g. *HTPheno_analysis_results_2010-07-07–15-41-26.csv*).

# Appendix A

# Details: workflow modules of HTPheno

The **HTPheno** plugin consists of several modules with defined interfaces that provide the current functionality (see Fig. A.1).
Please look at the javadoc documentation (and the source code) to obtain detailed information about modules and their interfaces.

**1** INPUT

- read image from database, file, etc. ...
- provides functions: **htpInputGetImage(); ...**

**2** CONFIGURATION

- load standard classification configuration files
- load user defined configuration files

**3** PROCESSING

- pattern recognition, classification of image areas, etc. ...

**4** CALCULATION

- current calculation of image parameters, e.g. number of green pixel, diameter of the plant, etc. ...

**5** OUTPUT

- export of the calculated data into *.csv result file, etc. ...

**Figure A.1:** Modules of **HTPheno**

# Appendix B

# Details: The current workflow in *HTP.java*

## B.1 Obtaining the image

An image is obtained via the *HTPinput.getNextImage()* function.

## B.2 Loading the configuration files

Depending on the image attribute (e.g. side view image or top view image) the configuration files are loaded.
There are five different files that contain configuration information for the image classification process:

- **region definitions** which specify image areas and a list of allowed classes
- **colour to class definitions** which declare the colour ranges (minimum and maximum in RGB and HSV) for each class
- **class to number definition** for the internal representation of each class as an integer number
- **number to colour definitions** defining the colour that each class will have in the classification image
- **scaling definitions** which allow to convert the measured pixel into millimetre.

## B.3 Colour range classification

For each defined region **HTPheno** conducts the classification based on the colour range. A pixel is classified as belonging to class c if its colour values in RGB and HSV fall within the range of the given values for class c. Classes are processed in the order given in the regions configuration files.

## B.4 Single pixel removal

As next step **HTPheno** removes isolated pixel that do not have at least 2 pixel of the same class within their neighbourhood of 8 pixel. This step lowers the noise level.

## B.5 Remove non connected clusters of green pixel

As the plant itself can be considered connected, all clusters of green pixel that are not within a certain radius near other plant parts, are discarded as incorrect segmented.

## B.6 Fill single pixel holes

For each still unclassified pixel we examine the pixel in a circle of radius 1 and 2 around that pixel. If only pixel of the same class lie on one of that circles, we also assign that class to our previously unclassified pixel.

## B.7 Apply region fill

All contiguous sets of unclassified pixel that are surrounded only by pixel of one class are also considered to be pixel of that class.

## B.8 Classify according to colour similarity

For an unclassified pixel all neighbouring classified pixel, and their classes, are noted acquired.

1. calculate the euclidean RGB and HSV distance to each of the neighbouring classes and determine the "nearest" class amongst these,

2. determine the most frequent class around that pixel,

3. decision: if the nearest and the most frequent class is of type *plant* then class *plant* is assigned; else the class with the closest match in colour values is assigned.

## B.9 Apply region fill again

All contiguous sets of unclassified pixel that are surrounded only by pixel of one class are also considered to be pixel of that class.

## B.10 Conduct image property calculations

Here **HTPheno** determines several image parameters like the width and height of the plant and the plant area and stores those values in the ImageJ results table for later display.

## B.11 Create the final image stack

Finally **HTPheno** creates the image stack that contains all images of the processing steps as well as the final image with contour, plant bounding box, scale and diameter, if applicable.