Short description of the centroiding algorithm and API

As in all image processing applications, the image from the camera runs through several image processing steps in a specific order to guarantee the correct results of computing the center of gravity. This document describes these steps and how they can be adapted by the Framegrabber API.

1. Setting the ROI (region of interest): The first step of the algorithm is to set the ROI, where the centroids will be computed. The size of the ROI can be changed in the API, but only values between 1 and 512 pixels will be accepted:

```
FgError_t FrameGrabber::setCentroidingRoi(const cv::Rect2i
&region_of_interest) {
    centroiding_roi_ = region_of_interest;
    centroiding_roi_.width = std::min(centroiding_roi_.width, 512);
    centroiding_roi_.height = std::min(centroiding_roi_.height, 512);
…
```

   If the input is bigger than 512 it will be clamped to 512. After the ROI is set, all processing steps will be only applied to the ROI! This saves a lot of resources.
2. Preprocessing the image: To ensure maximum accuracy when computing CoGs (center of gravity), the image has to be pre-processed. At first, a median filter with size 3x3 is applied to the image. This eliminates all hot pixels and reduces noise, which would lead to wrong CoGs. The median filter is an unchangeable processing step of the applet.
3. Thresholding the image: When all hot pixels are eliminated by the median filter, the maximum grey value of the ROI will be saved. This value is multiplied by a number between 0-1 (percentage) – every pixel underneath the result (max grey value * percentage) is set to 0, otherwise the grey values remain. The percentage of the maximum grey value can be changed in the API:

```
FgError_t FrameGrabber::setStarThresholding(float percentage_of_maximum){
    int percentage_shifted = percentage_of_maximum*255;
    if(Fg_setParameterWithType(fg_, LOWER_THRESHOLD_ID,
percentage_shifted, DMA_INDEX) != FG_OK) return
handle_fg_error_internal();
    return {};
}
```

4. CoGs: After thresholding the image, the centers of gravity (x, y) will be computed. The main concept is that the weighed sum (sum of all gray values multiplied by their coordinates) is divided by the total sum of the thresholded image grey values. This step is fixed in the applet and can't be adjusted in the API.
5. Rejection values: The API offers several rejection thresholds to prevent wrongly computed CoGs, etc. The applet provides the mean value before and after thresholding (sum of all grey values divided by the number of pixels before and after thresholding the image) as well as the maximum grey value and the signal to noise ratio (SNR, calculated by dividing max grey value by mean before threshold value) – all of these values are written to the end of the image.
    a. If the mean before thresholding value is too low the image is either underexposed or there is no guide star in the selected ROI, if the value is too

high the image is either overexposed or there is more than one guide star in the ROI (API: `current_image_info_.mean_before_threshold`).
The upper and lower rejection thresholds can be set. API:

```
void FrameGrabber::setMaximumMean(float maximum_mean) {
    maximum_mean_ = maximum_mean;
}
void FrameGrabber::setMinimumMean(float minimum_mean) {
    minimum_mean_ = minimum_mean;
}
```

b. If the mean after thresholding value is too high or too low, the centroiding will fail (API: `current_image_info_.mean_after_threshold`). The upper rejection threshold can be set. API:

```
void FrameGrabber::setMaximumMeanAfterThresholding(float
mean_after_thresholding) {
    maximum_mean_after_thresholding_ = mean_after_thresholding;
}
```

c. The SNR (API: `current_image_info_.signal_to_noise_ratio`) is a value to check image quality regarding noise. The lower rejection can be set. API:

```
void FrameGrabber::setMinimumSNR(float minimum_snr) {
    minimum_snr_ = minimum_snr;
}
```

d. The maximum grey value (`current_image_info_.max_grey_value`) is an indicator for either too many hot pixels (which couldn't be removed by median filtering) in the ROI or an overexposed image. API:

```
void FrameGrabber::setMaximumGreyValue(uint16_t max_grey_value) {
    maximum_grey_value_ = max_grey_value;
}
```

Important: If one of the listed rejection value is not set, the default values (see header file) will be used. The default values were developed during our first simulation and probably won't work for other simulation scenarios.