**Title:** *Method for Noise Filtering and Relocation of Optical Interferometry Images for Wear Analysis*

**Authors:** *A. G. Passos, T. Cousseau.*

**Affiliations:** *Mechanical Engineering Department, Federal University of Technology - Parana – Curitiba, PR, Brazil.*

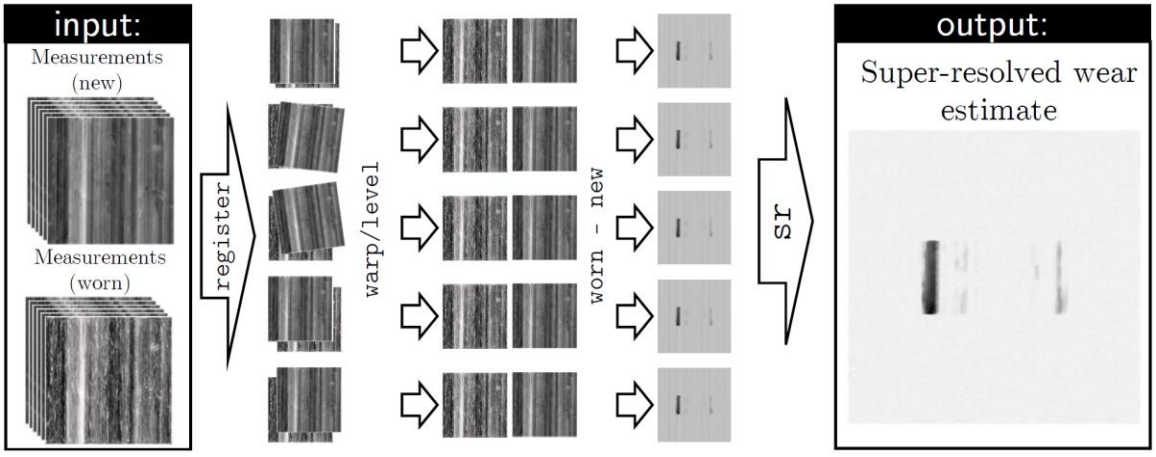
**Contact email:** *adrianogpassos@utfpr.edu.br*

**Abstract:** *Wear has always been an important issue in industry and academy. Its evaluation using optical interferometry assisted by image analysis is getting quite popular; however, it is susceptible to different sources of errors. In fact, when the wear to be evaluated is in the same order of magnitude of the asperities height, such errors prevent accurate quantification. To minimize this problem, a novel framework for almost zero wear evaluation is proposed. The approach is based on super-resolution and sub-pixel registration algorithms. Basically, from several measurements of new and worn surfaces, a sub-pixel registration algorithm is used to perform the surface relocation (warp and levelling). Then the relocated worn surfaces are subtracted from the new ones. As a result, several images of the wear are obtained. All these images are then processed in an algorithm of super resolution, which lead to the super-resolved wear. The highlights of this technique are presented below:*

- *The results presented suggest that the technique could meet the demands of most current tribology almost zero wear evaluation.*
- *In the benchmark test proposed by this paper a significant improvement was observed when compared to a standard approach.*
- *The developed software is open source and available online.*

**Keywords:** *Registration; Denoising; Super-Resolution; Wear*

**Graphical abstract:**



**Specifications Table**

| | |
|---|---|
| Subject area | *Engineering* |
| More specific subject area | *Tribology* |
| Method name | *Method for Noise Filtering and Relocation of Optical Interferometry Images for Wear Analysis* |
| Name and reference of original method | - |
| Resource availability | *All links are provided on the text for better context.* |

## Method details

Wear evaluation using optical interferometry assisted by image analysis is susceptible to different sources of errors. Here we provide a robust procedure to estimate "almost zero wear" of engineering surfaces with minimum error. In order to allow for this work to be reproduced, a detailed explanation on how to apply the proposed method on synthetic data is provided.

The procedure to generate the artificial data-set of simulated "measurements" used to validate the framework (proposed on the Part 2 of the main paper) is shown in Figure 1. These surfaces were obtained from a single measurement (real measurement), which was levelled and filtered to be simulate the real surface (*true* new surface). A threshold was applied to the *true* new surface to simulate wear, this surface is called the *true* worn surface. Then, *true* new and worn surfaces were degraded multiple times independently by the addition of random noise and motion to simulate the errors added by the measurements. Finally the proposed algorithm was applied to estimate the true wear surface.

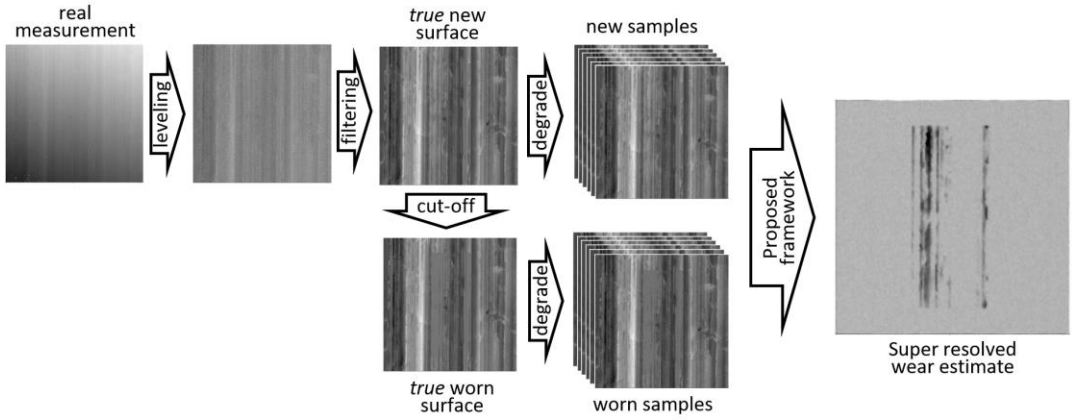



*Figure 1: Block-diagram of the validation method.*

## Installing and loading the required software:

Download and install R from https://cran.r-project.org/mirrors.html and Rstudio IDE (optional) from https://www.rstudio.com/products/rstudio/download/. Both softwares are available for free at those pages.

All the following commands can be run on a R console. To install and load the redR package simply run:

```
install.packages('redR')
install.packages('purrr')
library(redR)
library(purrr)
```

**Surface Simulation**

In order to simulate real measurements the following procedures is used

**Reading a real measurement file**

First we read a sample cloud point `txt` file, exported by the MountainsMap Imaging Topography Software, from the github website. The data was exported in a raw format in order to avoid unknown image processing:

```
file = 'https://raw.githubusercontent.com/coldfir3/RED/master/measurement
.txt'

# reading the txt file and forcing the sorage mode to be numeric
# this is needed due the fact that missing values are saved as '***'
# by MountainsMap by default
file %>%
  read.table %>%
  as.matrix %>%
  unname %>%
  as.numeric %>%
  as.cimg -> im
```

Alternatively a local file could be read by changing the `file` variable to the complete path of the measurement file. For example

```
file = 'C:/measurements/measurement1.txt'
```

Since the original raw data have missing values (usually the case on any real measurement) we replace them by the median value of closest 8 neighbors:

```
# replacing missing values at the borders with the image median value
pxs <- px.borders(im, 2) & is.na(im)
im[pxs] <- stats::median(im, na.rm = TRUE)

# replacing all other missing values with median value of closest neighbo
urs
pxs <- !px.borders(im, 2) & is.na(im)
im[pxs] <- where(pxs) %>%
  pmap(
    function(x, y, ...){
      imager::get.stencil(
        im = im,
```

```
        stencil = expand.grid(dx=seq(-2,2,1),dy=seq(-2,2,1)),
        x = x,
        y = y)
      }
    ) %>%
  map_dbl(stats::median, na.rm = TRUE)
```

The raw data was levelled and filtered using the following code:

```
# leveling the measurement by linear regression and then removing some no
ise
true_new_sample <- im %>%
  (function(x) x - stats::lm(value ~ x + y, data = as.data.frame(x)) %>%
fitted) %>%
  RED(lambda = 5, functional = 'DN', niter = 5)
```

The filtering was applied to simulate a noise-free physical surface (*true* new surface). This is required because the measured surface coitains random noise. The resulting image is considered exactly as the physical surface (unaffected by any measurement defect).

From the *true* new surface, a *true* worn surface is created by applying a threshold cut on a selected area (simulating a tribologic test)

```
threshold = 0.2
# create the mask of the allowable wear area
mask <- (
  px.top(true_new_sample, 56*2) |
  px.bottom(true_new_sample, 56*2) |
  px.left(true_new_sample, 56*6) |
  px.right(true_new_sample, 56*6)
)
# create the mask of the wear: pixel heights inside the allowable
# area that are higher than the threshold
mask2 <- true_new_sample > threshold & !mask

# apply the threshold value to the worn pixels
true_worn_sample <- true_new_sample * (mask | !mask2) + threshold * mask2
# create an TRUE wear surface for post processing
true_wear_sample <- true_worn_sample - true_new_sample
```

### Degradation model

From the *true* new and worn surfaces a set of imperfect measurements are simulated based on the following degradation model.

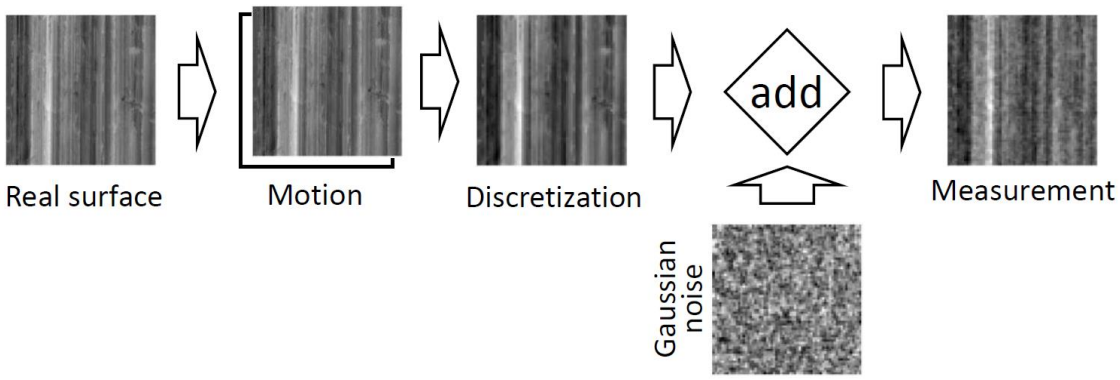



*Figure 2: Degradation model used on the present work.*

The code to generate a simulated measurement based on an *true* input image, a noise level and a range of allowable random motion (in pixels) is presented as follows:

```
simulate <- function(im, noise, motion){

  # select random relocation parameters
  s = rbind(sample(-motion:motion, 2))

  # degrade the samples using the redR function 'degrade'
  # after degradation the mean of the surface is subtracted from
  # each pixel to mimic the true measuring process.
  degraded_im <- redR::degrade(im, L = 4, s = s, noise = noise)
  degraded_im <- degraded_im - mean(degraded_im)

  # return the degraded image
  return(degraded_im)
}
```

A single measurement procedure can be simulated by using

```
new_sample <- simulate(im = true_new_sample, noise = 0.1, motion = 4)
worn_sample <- simulate(im = true_worn_sample, noise = 0.1, motion = 4)
wear_sample <- new_sample - worn_sample
plot(imlist(new = new_sample, worn = worn_sample, wear = wear_sample), ax
es = FALSE, layout = 'row')
```

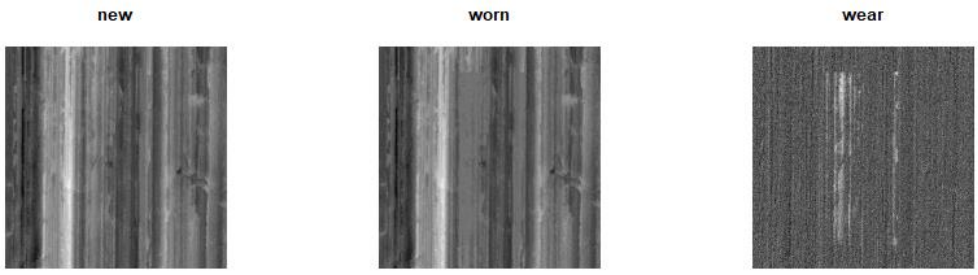


*Figure 3: Simulated measurements*

### Generating a set of simulated measurements

Here, we generate 5 new and 5 worn measurements. This represents 5 sequential measurements of the new sample followed by some tribological test and then 5 sequential measurements of the worn sample. For better results we advise the usage of at least 20 measurements for each new and worn surfaces.

```
new_samples <- replicate(5, simulate(im = true_new_sample, noise = 0.1, m
otion = 4), simplify = FALSE) %>% as.imlist
worn_samples <- replicate(5, simulate(im = true_worn_sample, noise = 0.1,
motion = 4), simplify = FALSE) %>% as.imlist
```

### Super-resolving the wear image

To obtain the wear surface data we apply the following algorithm

```
# estimate the initial in-plane registration parameters
ip_par <- map2(new_samples, worn_samples, redR::register, method = 'taylo
r3', tol = 1e-5, maxiter = 100)

## do the in-plane relocation and estimate the out-of-plane registration
parameters
new_samples_star <- map2(new_samples, ip_par, transform) %>% as.imlist
wear <- map2(worn_samples, new_samples_star, `-`) %>% as.imlist
mask <- (
  px.top(wear[[1]], 56*2/4) |
  px.bottom(wear[[1]], 56*2/4) |
  px.left(wear[[1]], 56*6/4) |
  px.right(wear[[1]], 56*6/4)
  ) %>%
  as.numeric %>%
  as.vector
model <- map(wear, function(data) stats::lm(value ~ x + y, data = as.data
.frame(data), weights = mask))

# do the out-of-plane relocation and update wear estimates
new_samples_star <- map2(new_samples_star, (model %>% map(fitted)), `+`)
%>% as.imlist
wear <- map2(worn_samples, new_samples_star, `-`) %>% as.imlist

# estimate in-plane inter-wear relocation parameters (relocate all wear e
stimates in relation to the first one)
ip_par_wear <- map(wear, redR::register, wear[[1]], method = 'taylor3', t
ol = 1e-5, maxiter = 100) %>%
  do.call(rbind, .)

# super resolve the wear estimation using all low-res wear estimations
wear_HR <- RED(imappend(wear, 'z'), lambda = 10, functional = 'SR', args
= list(scale = 4, s = ip_par_wear))
```

The algorithm's outputs are presented as follow:

```
results <- imlist(
  'true' = true_wear_sample,
  'cubic-interpolated' = imager::imresize(wear[[3]], 4, 5),
  'super-resolved' = wear_HR
  )
plot(results, axes = FALSE, layout = 'row', interpolate = FALSE)
```

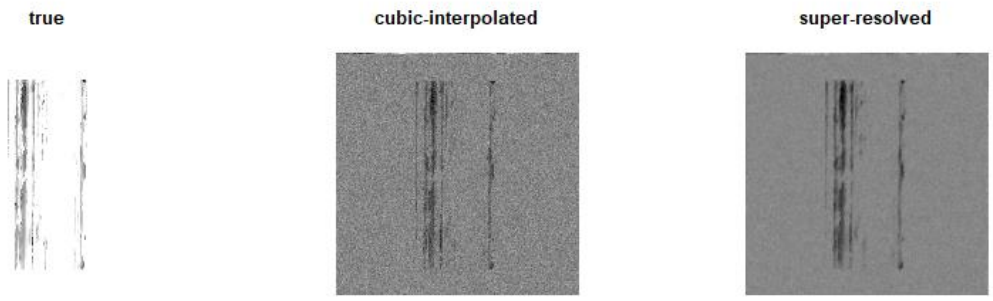



*Figure 4: Comparative results*

```
# Mean Square Error:
results %>% map_dbl(redR::MSE, results[[1]])

##               true cubic-interpolated     super-resolved
##        0.000000000        0.022125272        0.003966882

# Mean Absolute Error:
results %>% map_dbl(redR::MAE, results[[1]])

##               true cubic-interpolated     super-resolved
##         0.00000000         0.11381480         0.03603793
```