Photometric stereo on the Web thanks to Rust and Wasm

Matthieu Pizenberg

Bayeux museum

Normandie Université

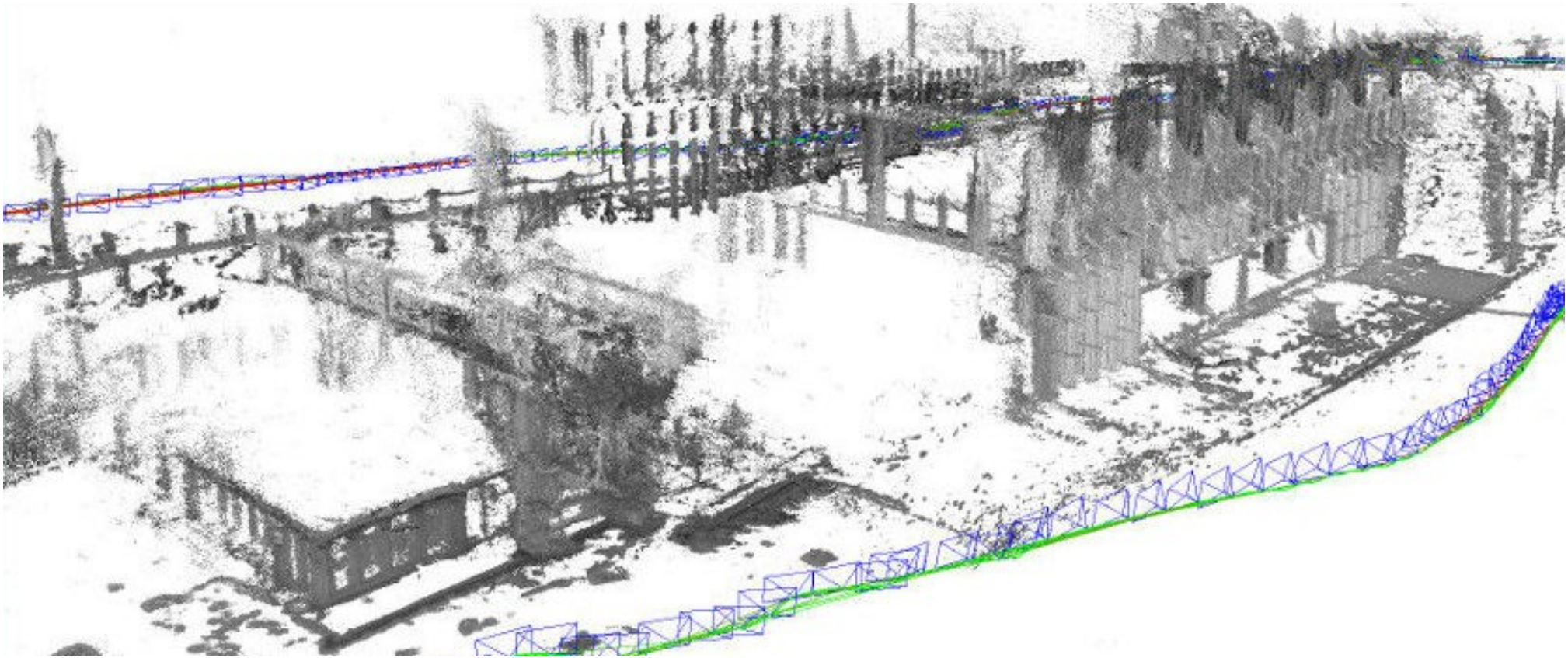
PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR

*** 3D Reconstruction ***

Photometric Stereo

Porting to the Web with Rust

3D Reconstruction from images

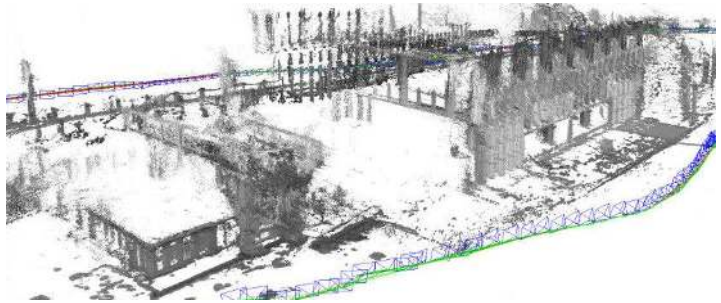


LSD-SLAM, Engel et al., ECCV 2014

3D Reconstruction from images

Kinect (1 image, structured light / ToF)

Photogrammetry / SfM / vSLAM
(multiple images, moving camera,
stable lighting conditions)



LSD-SLAM: Large-Scale Direct Monocular SLAM, Engels et al., ECCV 2014

Photometric Stereo
(multiple images, fixed camera,
variable lighting)

Stéréophotométrie microscopique sans démosaïquage, Quéau et al. , GRETSI 2017

3D Reconstruction from images

Photometric Stereo
(multiple images, fixed camera,
variable lighting)

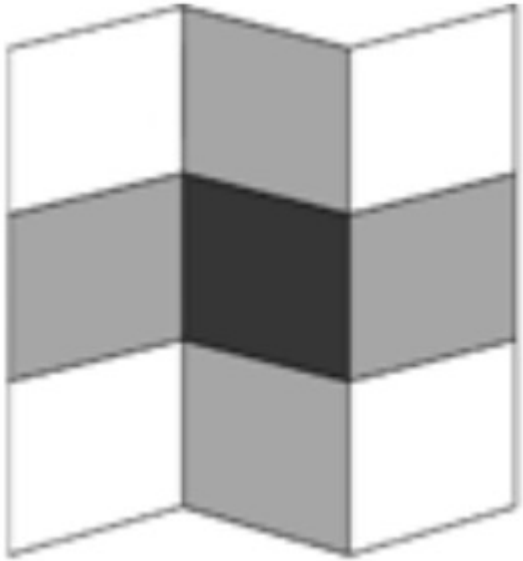
Stéréophotométrie microscopique sans démosaïquage, Quéau et al. , GRETSI 2017

3D Reconstruction

*** **Photometric Stereo** ***

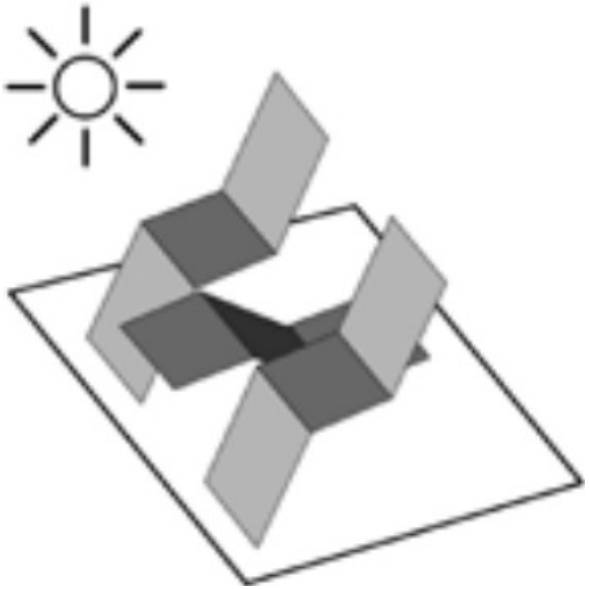
Porting to the Web with Rust

Perception of shading and reflectance



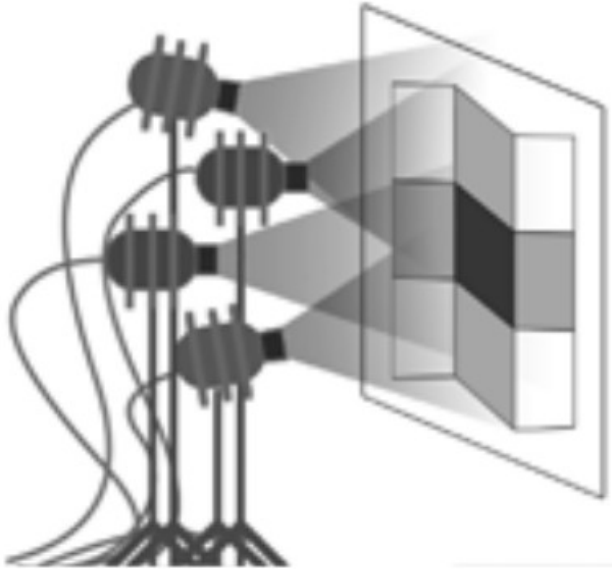
How to interpret this image ?

Perception of shading and reflectance



Explanation of the sculptor,

of the painter,



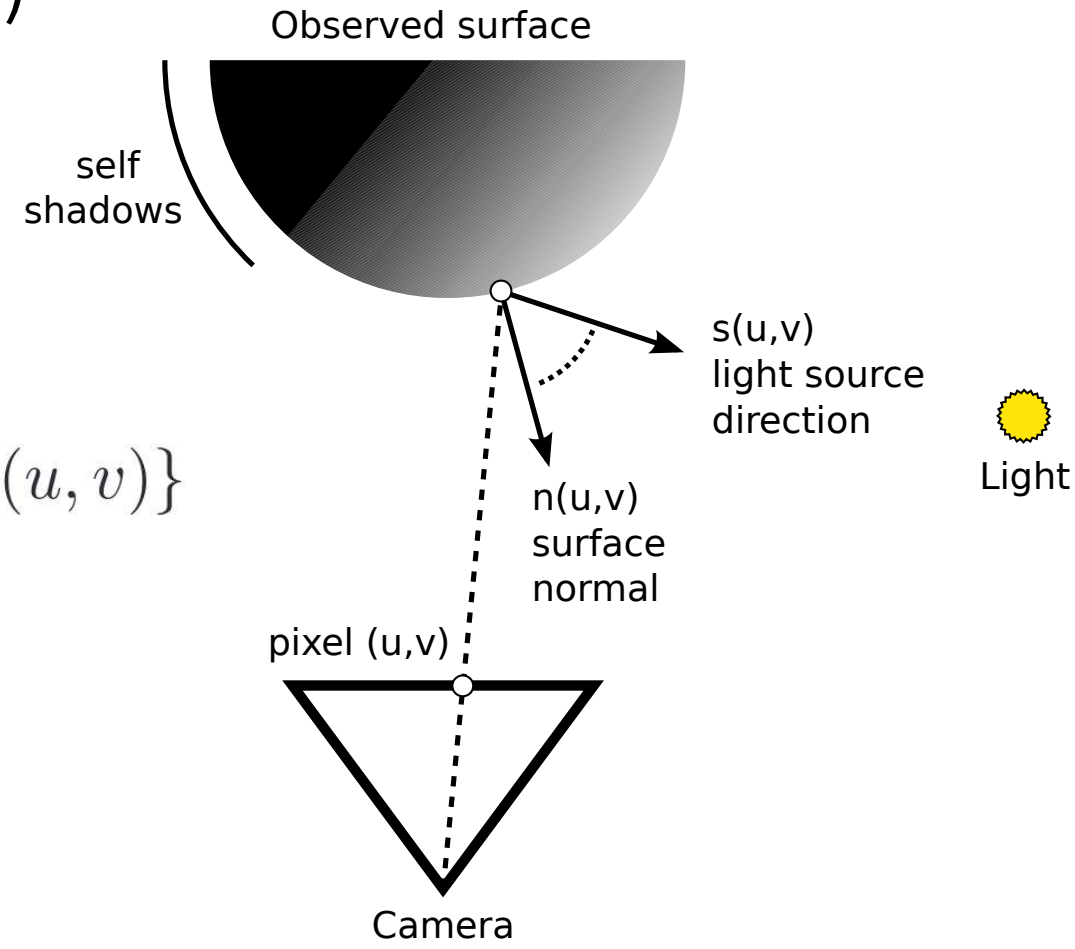
of the lighting designer.

Adelson and Pentland's workshop metaphor, The perception of shading and reflectance, 1996

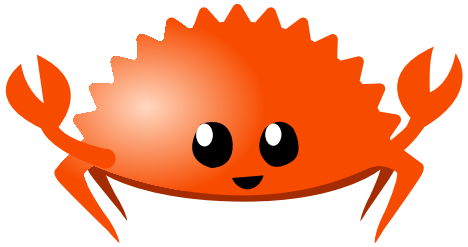
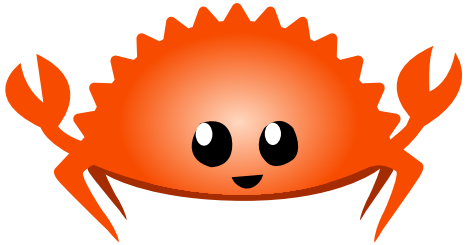
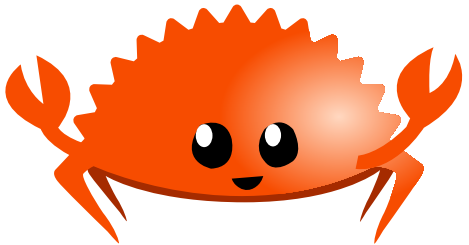
Shape from Shading (SfS)

Lambert's law :

$$I^1(u,v) = \rho(u,v)\max\{0, \mathbf{s}^1 \cdot \mathbf{n}(u,v)\}$$



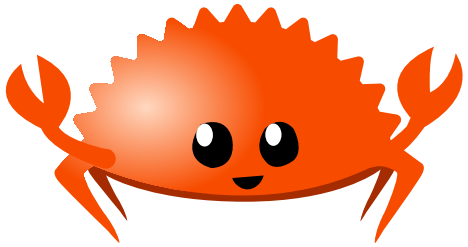
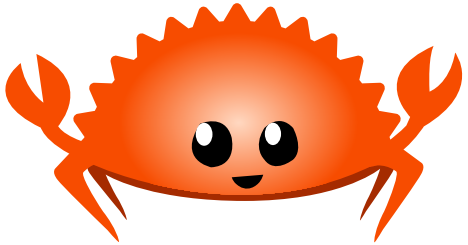
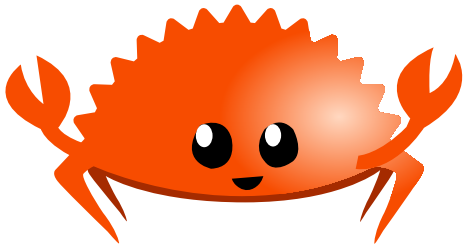
Photometric Stereo

Let’s take multiple images instead of one, with different lighting for each.

$$\begin{cases} I^1(u,v) = \rho(u,v)\max\{0, \mathbf{s}^1 \cdot \mathbf{n}(u,v)\} \\ \vdots \\ I^m(u,v) = \rho(u,v)\max\{0, \mathbf{s}^m \cdot \mathbf{n}(u,v)\} \end{cases}$$

Photometric Stereo

Let’s take multiple images instead of one, with different lighting for each.

$$\underbrace{\begin{bmatrix} I^1(u,v) \\ \vdots \\ I^m(u,v) \end{bmatrix}}_{\mathbf{i}(u,v) \in \mathbb{R}^m} = \underbrace{\begin{bmatrix} \mathbf{s}^{1\top} \\ \vdots \\ \mathbf{s}^{m\top} \end{bmatrix}}_{\mathbf{S} \in \mathbb{R}^{m \times 3}} \underbrace{\begin{bmatrix} \rho(u,v)\mathbf{n}(u,v) \end{bmatrix}}_{\mathbf{m}(u,v) \in \mathbb{R}^3}$$

Photometric Stereo

1 euro (Italy)

50 cents (Spain)

1 yuan (China)

3D-reconstructions

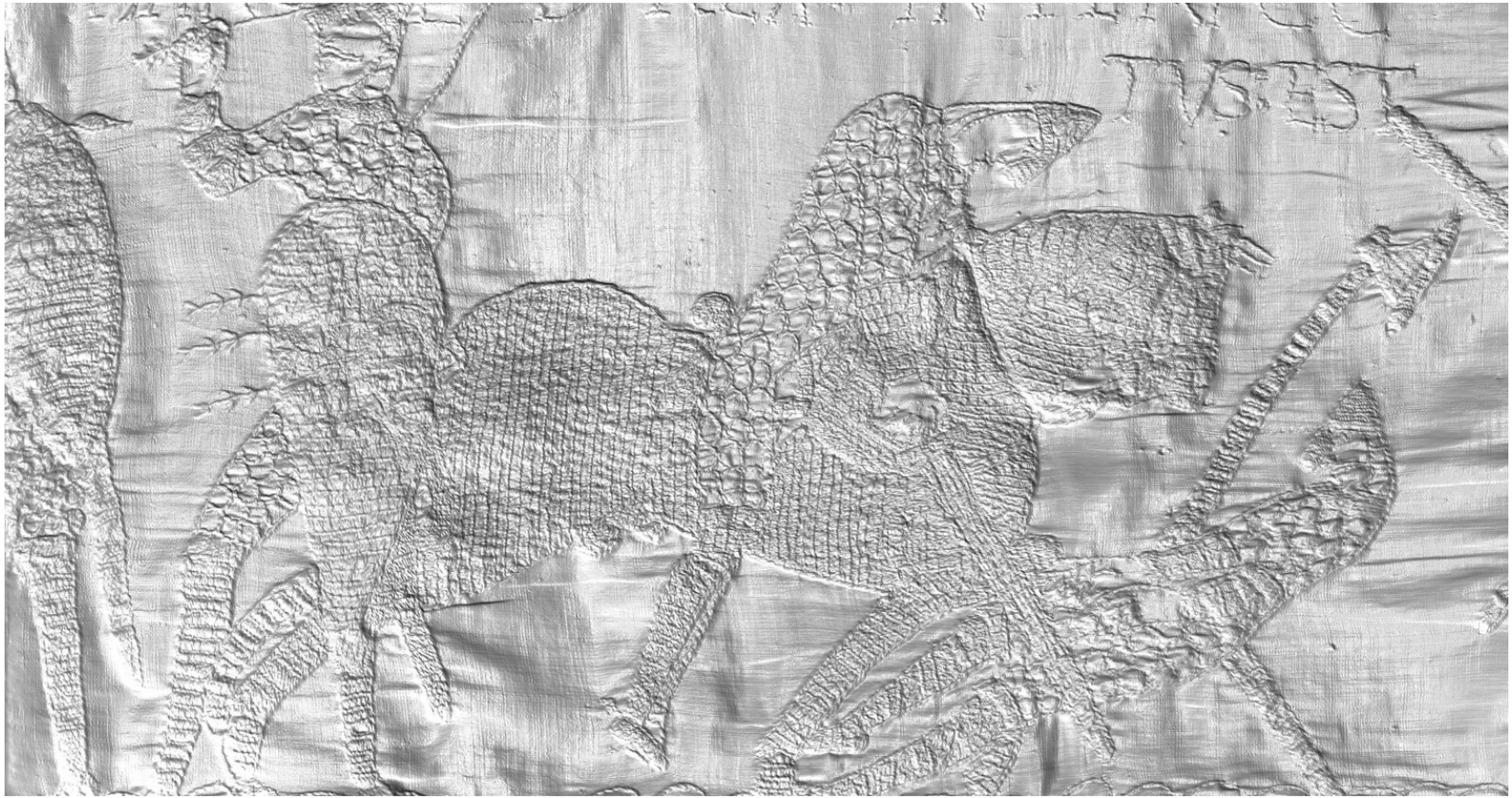
Photometric Stereo – captation (Bayeux Tapestry)

Photometric Stereo – Bayeux Tapestry

Photometric Stereo – Bayeux Tapestry

Photometric Stereo – Bayeux Tapestry

Photometric Stereo – Bayeux Tapestry



3D Reconstruction

Photometric Stereo

***** Porting to the Web with Rust *****

Demo

Code organization

Code organized in 4 main directories :

- cal-qp-ps-lib/: the core parts of the algorithm presented as a Rust library.
- cal-qp-ps-bin/: an example CLI executable program.
- cal-qp-ps-wasm/: the WebAssembly modules exposing the algorithm in wasm.
- web-elm/: the frontend application, made in Elm.

Library code

```rust
type Mat3D = DMatrix<(f32, f32, f32)>;

/// Configuration (parameters) of the photometric stereo algorithm.
pub struct Config {
    pub max_iterations: usize,
    pub threshold: f32,
    pub z_mean: f32,
    pub lights: Vec<(f32, f32, f32)>,
}

/// Compute the depth, normals and albedo from a sequence of images
/// with different lighting conditions.
/// Returns (xyz, normals, albedo)
pub fn photometric_stereo(
    config: Config,
    raw_images: &[DMatrix<f32>], // f32 in [0,1]
) -> (Mat3D, Mat3D, DMatrix<f32>) { ... }
```

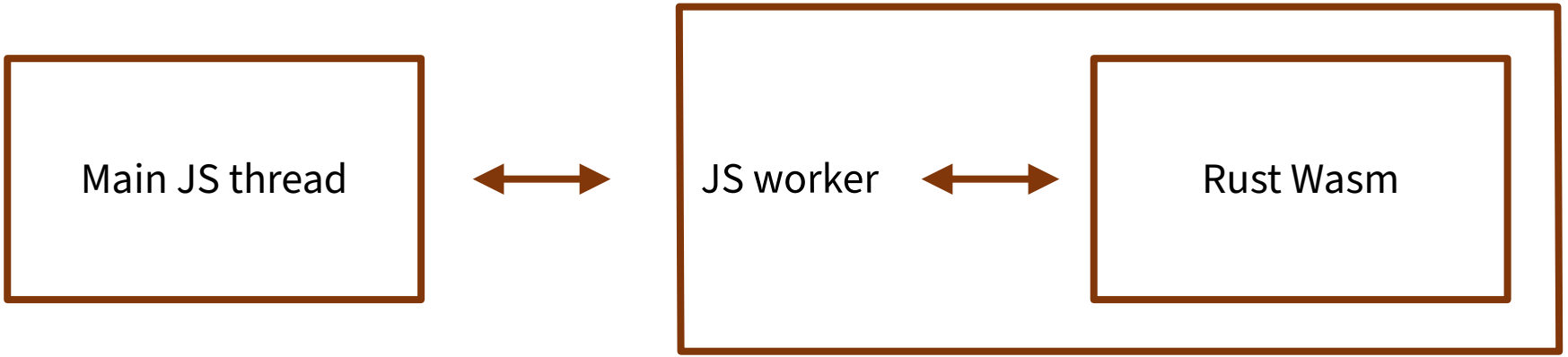
Library config

```toml
[package]
name = "cal-qp-ps-lib"
version = "0.1.0"
authors = ["Matthieu Pizenberg <matthieu.pizenberg@gmail.com>"]
edition = "2018"

[dependencies]
nalgebra = "0.25.1"
image = { version = "0.23.14", default-features = false }
thiserror = "1.0.29"
wasm-bindgen = { version = "0.2.78", optional = true }
serde = { version = "1.0.130", optional = true }
```

Rust WebAssembly config

```toml
[package]
...

[lib]
crate-type = ["cdylib", "rlib"]

[dependencies]
cal-qp-ps-lib = { path = "../cal-qp-ps-lib", features = ["wasm-bindgen", "serde"] }
image = { version = "0.23.14", default-features = false, features = ["jpeg", "png"] }
wasm-bindgen = { version = "0.2.78", features = ["serde-serialize"] }
js-sys = "0.3.55"
...

# For better error messages when panicking
console_error_panic_hook = { version = "0.1.6", optional = true }

[profile.release]
opt-level = "s" # optimize for small code size.
```

Rust ↔ Wasm ↔ JS



Rust WebAssembly code

cal-qp-ps-wasm/src/lib.rs

```rust
// Wrapper trick since we cannot have async functions referencing &self.
// https://github.com/rustwasm/wasm-bindgen/issues/1858
#[wasm_bindgen]
pub struct Algorithm(Rc<RefCell<AlgoInner>>);

#[wasm_bindgen]
impl Algorithm {
    pub fn init() -> Self {
        Algorithm(Rc::new(RefCell::new(AlgoInner::init())))
    }
    ...
}

// Store as fields all data that needs to be transferred
struct AlgoInner {
    image_ids: Vec<String>,
    dataset: Dataset,
    lights: Vec<(f32, f32, f32)>,
    normal_map: Vec<u8>,
}
```

JS worker calling WebAssembly

web-elm/static/worker.mjs

```
// Remark: ES modules are not supported in Web Workers
// esbuild worker.mjs --bundle --preserve-symlinks --outfile=worker.js
import { Algorithm as AlgoWasm, default as init } from "./pkg/cal_qp_ps_wasm.js";

// Initialize the wasm module.
let Algorithm;
(async function () {
  await init("./pkg/cal_qp_ps_wasm_bg.wasm");
  Algorithm = AlgoWasm.init();
})();

// Listener for messages
onmessage = async function (event) {
  if (event.data.type == "run") {
    await run(event.data.data);
  } else { ... }
};

// Main algorithm with the parameters passed as arguments.
async function run(params) {
  ...
  postMessage(..., await Algorithm.run(args));
}
```

Thank you !

- The code : https://github.com/mpizenberg/calibrated-quasi-planar-photometric-stereo
- Initial matlab algorithm from Yvain Quéau
- Web application made in collaboration with Florian Vincent