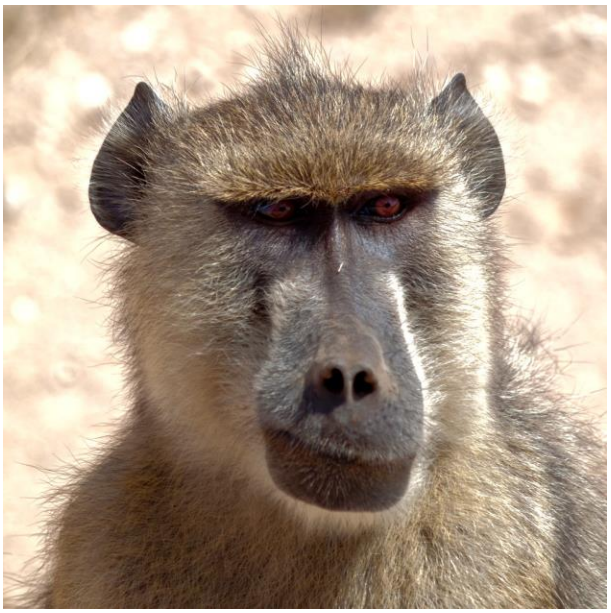

Photo by Luca Esposti on Flickr, CC-BY, adopted (with minor modifications) as The Shimmia.
This is a very serious code monkey.
It looks at your herd of data types with slight contempt, dead serious to bring it to order.

Shimmia package

The "shimmia" package name is a word play on "shim" (they call so a compatibility layer in software engineering) and "scimmia" (an ape or a monkey in Italian — this is a 'code monkey' level library, nothing smart at all). Of course, it has a backronym: SHIM between MinImage and Anything else. Or you may think of it as a derivative of "shimmin ← minshim", since it's a shim for min, but is not of itself a min-grade package.

So, its aim is to greatly simplify using min-anyting for interfaces and allow its use for bridging disparate platforms, like ROS, Qt, OpenCV and Eigen via Min*-approved types, therefore accelerating adoption of min-libraries by lowering entrance barriers and providing tangible advantages for using min* in multiplatform programming.

Design considerations

- Act like a monkey, be agile and mimic anything, though be aggressive and explicit and throw rotten fruit or f-words at bystanders if they behave poorly
- This is no gorilla — being lightweight is important, so no unnecessary dependencies, and preferably be header-only if this wouldn't conflict with other dependencies
- Clearly separate lossless and lossy conversions by naming conventions
- Be simple, no heavy template, macro or pointer magic; better to equip the user with simple sharp tools to make the magic themselves than to be responsible for cutting off their fingers
- Do not duplicate any minsubsystem functionality, but maybe provide a more convenient interface *for better integration with other programming styles*.

Interface conventions

- Header files live under the path shimmia/‹platform›/‹domain›.hpp, like: shimmia/ros/geometry_msgs.hpp (as ROS has several geometry type sets with different) or shimmia/ocv/geometry_msgs.hpp
- A projected set of headers therefore looks like this:
    - qt/geometry.hpp
    - qt/image.hpp
    - ocv/geometry.hpp
    - ocv/image.hpp
    - ros/geometry_msgs.hpp

    - ros/tf2.hpp
    - ros/image.hpp
    - eigen/geometry.hpp
    - boost/geometry.hpp — boost::geometry adapters
    - boost/gil.hpp — Generic Image Library adapters, very unlikely to happen
    - leptonica/image.h
    - leptonica/geometry.h
    - std/exceptions.hpp — THROW_ON_MINERR and MINERR_ON_EXCEPTION
    - std/image.hpp — MinImg over std vector, array, valarray and scalar
    - Maybe MKL and IPP should be included later
    - Would be nice to support OpenGL, OpenCL, Cuda, and AGG (the graphics library).
- Namespaces look like this:
    - shimmia::‹platform› — for top-level functions
- Function naming conventions:
    - Result as_‹generalized_type_name›(Input const&) — lossless conversions
    - Result to_‹generalized_type_name›(Input const&) — lossy conversions, truncating
    - Result round_to_‹generalized_type_name›(Input const&) — lossy conversions with proper rounding (as in static_cast<type>(input + 0.5))
    - Result ‹in_logic_type›_to_‹generalized_type_name›(Input const&) — conversions with ambiguous interpretation of input type
    - void {as|to}_‹generalized_type_name›(Output &, Input const&) — out arg version, result keeps to the left
    - copy_{as|to}_‹generalized_type_name› — for deep copy of heavy (variable-sized) data types
- Use std exceptions, not error codes, and throw aggressively
- Use C++11; limited C++03_TR1 compatibility may be implemented if requested
- Limited C compatibility may be implemented in separate files if required