**Domain-Specific Language**

DSL construct realized as compiler-known C++ classes:
- images & image pyramids
- filter mask / domain
- accessor
  - boundary handling
  - interpolation
- iteration space
- kernel

**Architecture Model**

Models hardware constraints:
- registers
- shared memory
- thread pool

Determines:
- kernel configuration (tiling)
- target-specific optimizations

**Runtime Support**

Manage available resources:
- device initialization
- context management
- kernel compilation
- synchronization

Utilize hardware features:
- unified memory support
- streaming

**Image Manipulation**

Computations on images are described in the `kernel()` and `reduce()` methods of user-defined C++ classes:
- implicit (relative) pixel access
- convolutions as C++11 lambda-function

**Optimizations**

Context-specific:
- MPMD code generation
- loop unrolling
- constant propagation

Target-specific:
- memory padding
- thread-coarsening
- memory hierarchy mapping
- vectorization

**Results**

High productivity:
- concise and compact algorithm description

Portability:
- cross-platform support from the same high-level description

Competitive performance:
- faster than other image processing DSLs and libraries