Recognition Pipeline and Object Detection Scalability

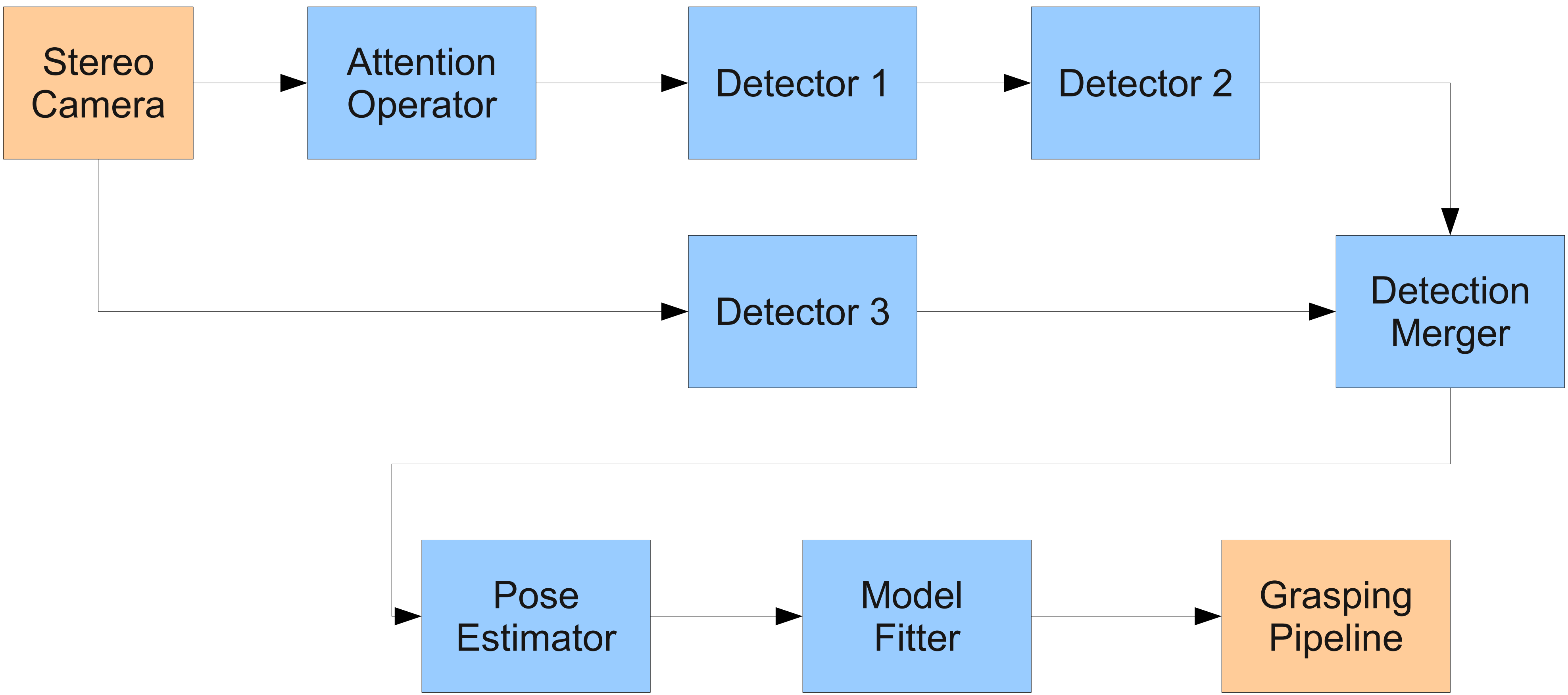
Marius Muja
University of British Columbia

Summer 2010 Internship Presentation

Recognition Pipeline

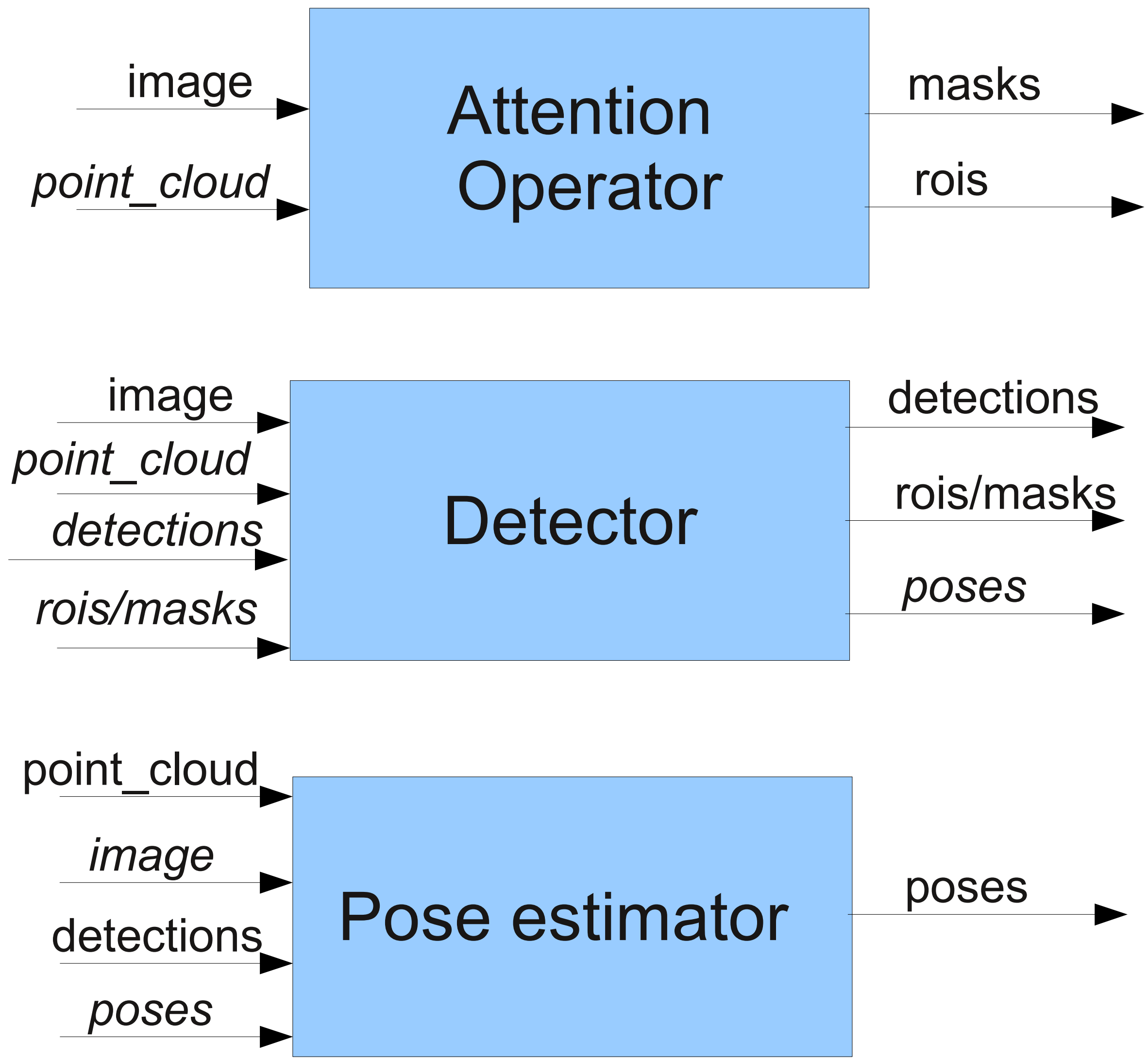
- Motivation
  - Easy to use vision algorithms without actually writing vision code
  - Easy to write vision algorithms without much knowledge of the rest of the system
  - “Plug and play”, swappable vision algorithms
  - Each algorithm is a “building block” that consumes *input(s)*, produces some *output(s)* and is configured by a set of *parameters*
    - Implemented as **nodelets** for efficiency reasons

Example



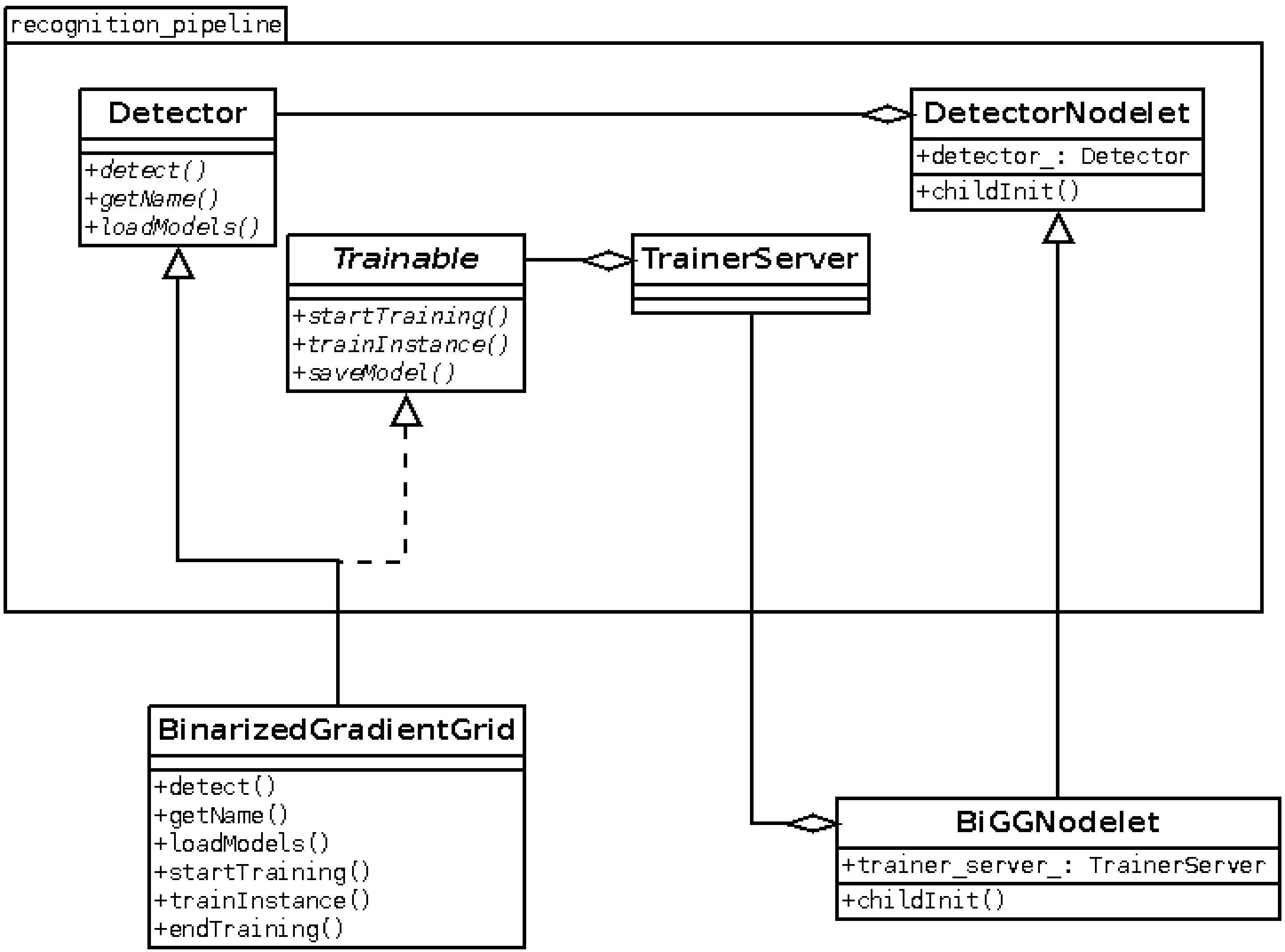
Stereo Camera
Attention Operator
Detector 1
Detector 2
Detector 3
Detection Merger
Pose Estimator
Model Fitter
Grasping Pipeline

Recognition Pipeline Components



Example

- Adding a new object detector to the recognition pipeline

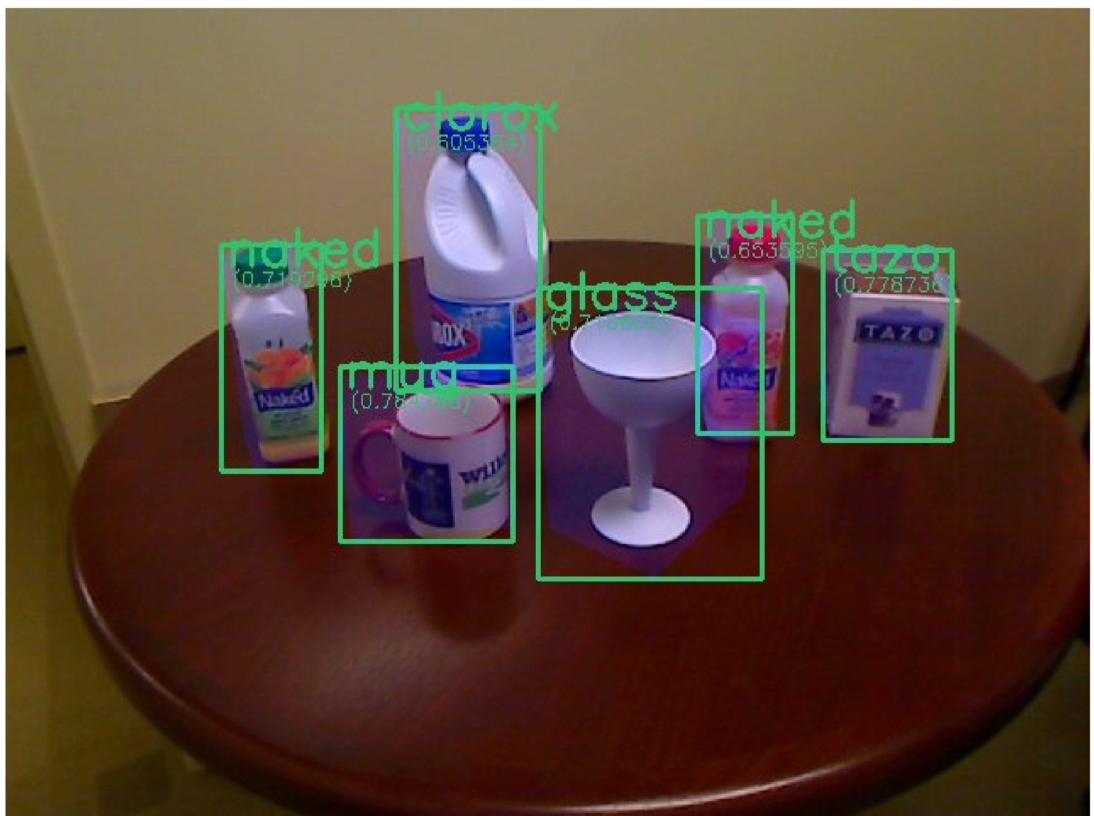


Recognition Pipeline Design

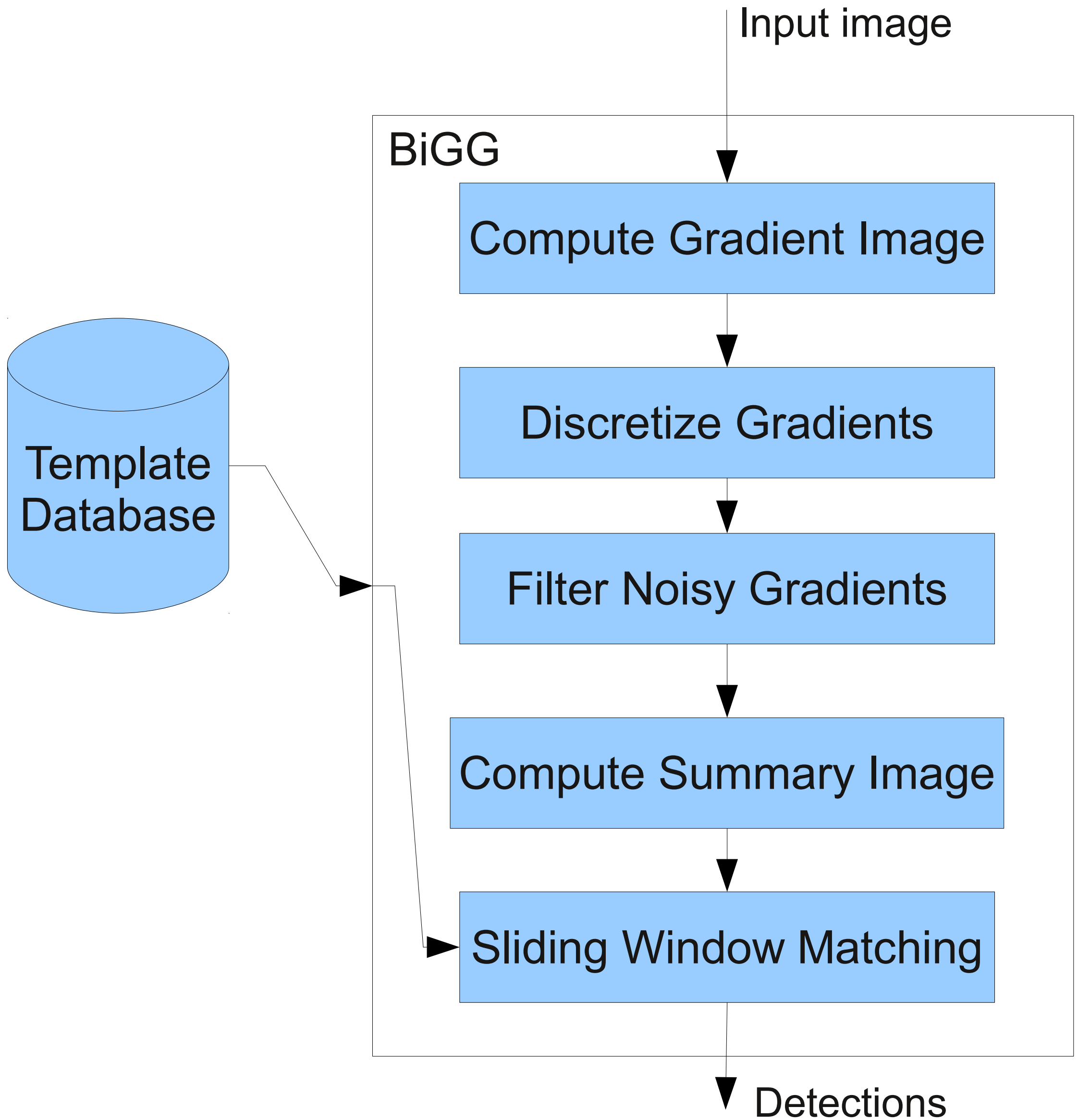
- Components
  - ROS-independent vision algorithms
  - ROS-wrappers for those algorithms (nodelets)
- Features
  - Easy to include additional algorithms as plugins (dynamically loadable/unloadable)
  - Build-in model persistence (currently using postgresql and sqlite3 databases or the file system)
  - TrainerServer – framework and GUI for training new models
- Located in the '`recognition_pipeline`' package

Binarized Gradient Grids (BiGG)

- **Goal**: fast and scalable object detection for rigid, non-articulated objects
- A template based object detection method



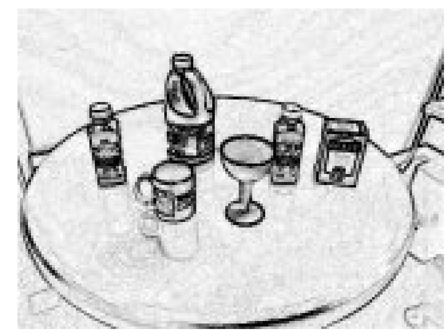
BiGG – Algorithm Outline



BiGG – Algorithm Outline

Input image
↓
Compute Gradient Image
↓
Discretize Gradients
↓
Filter Noisy Gradients
↓
Compute Summary Image
↓
Sliding Window Matching
↓
Detections

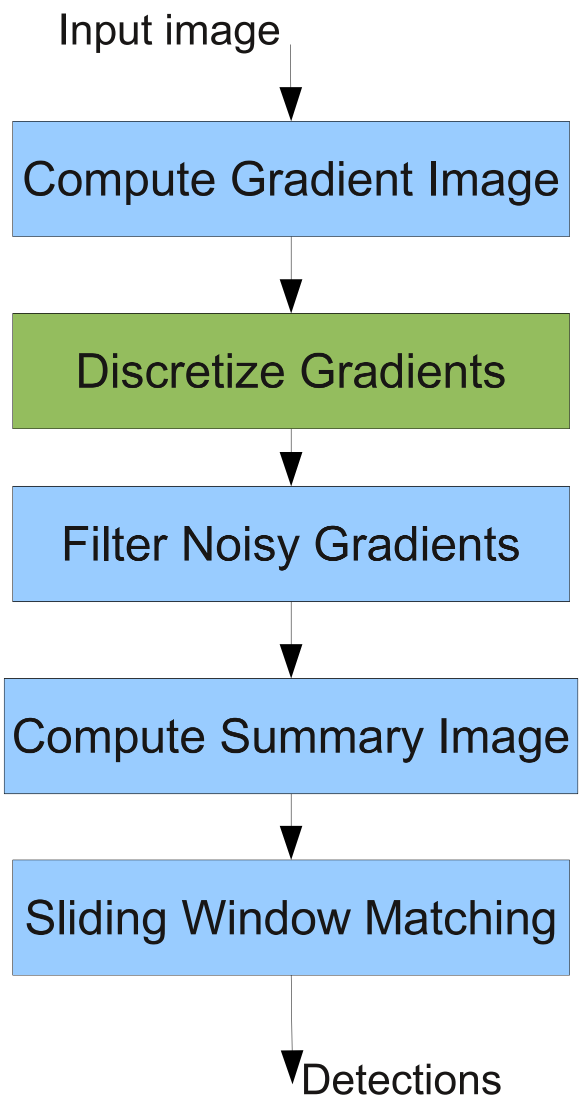
- Using gradient information instead of pixel values to be more robust to illumination changes



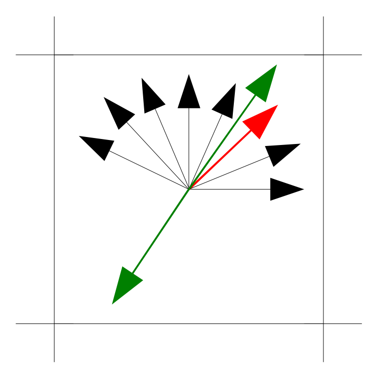
Magnitude

Orientation

BiGG – Algorithm Outline



- Discretize each gradient orientation into 8 bins
  - Use only orientation information to be more robust to contrast changes
  - Makes the algorithm robust to slight rotation changes
  - Use bit operations for fast matching
  - Ignore polarity of orientation



BiGG – Algorithm Outline

Input image

Compute Gradient Image

Discretize Gradients

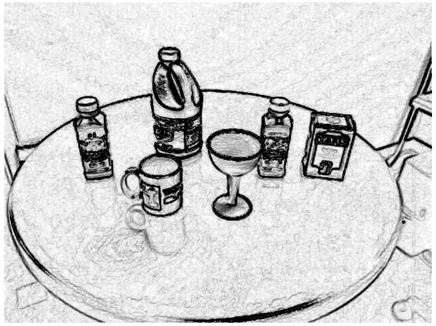
Filter Noisy Gradients

Compute Summary Image

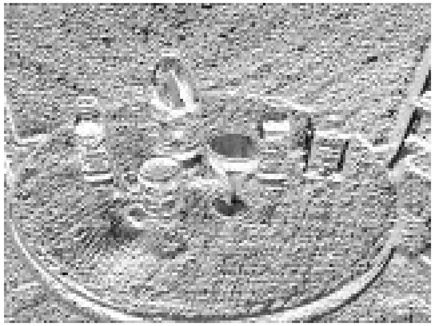
Sliding Window Matching

Detections

- Only use pixels with the magnitude above a certain threshold to be robust to noise



Magnitude

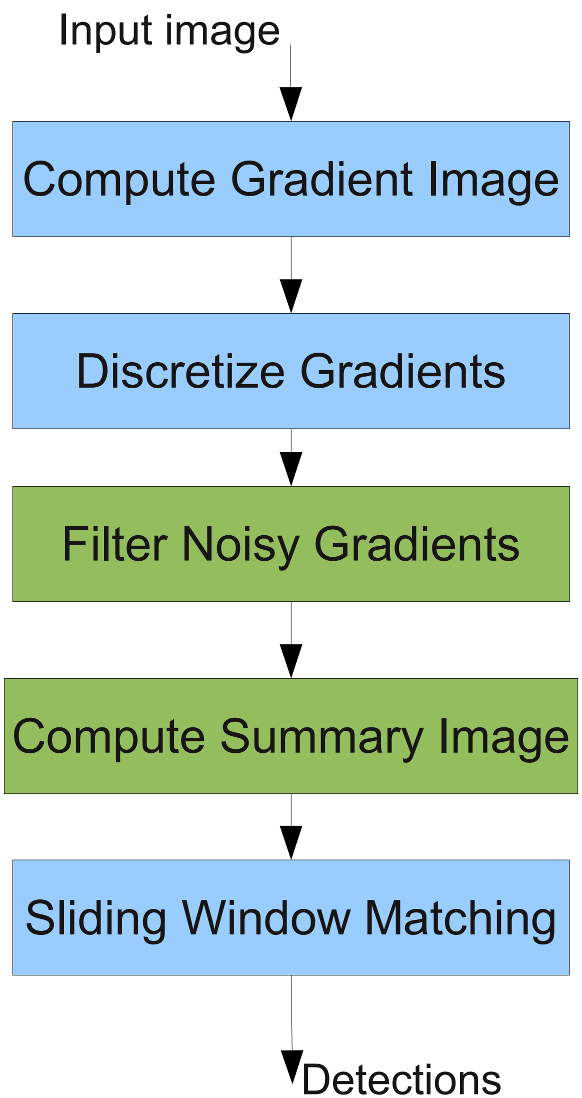


Orientation

Discretized Orientation

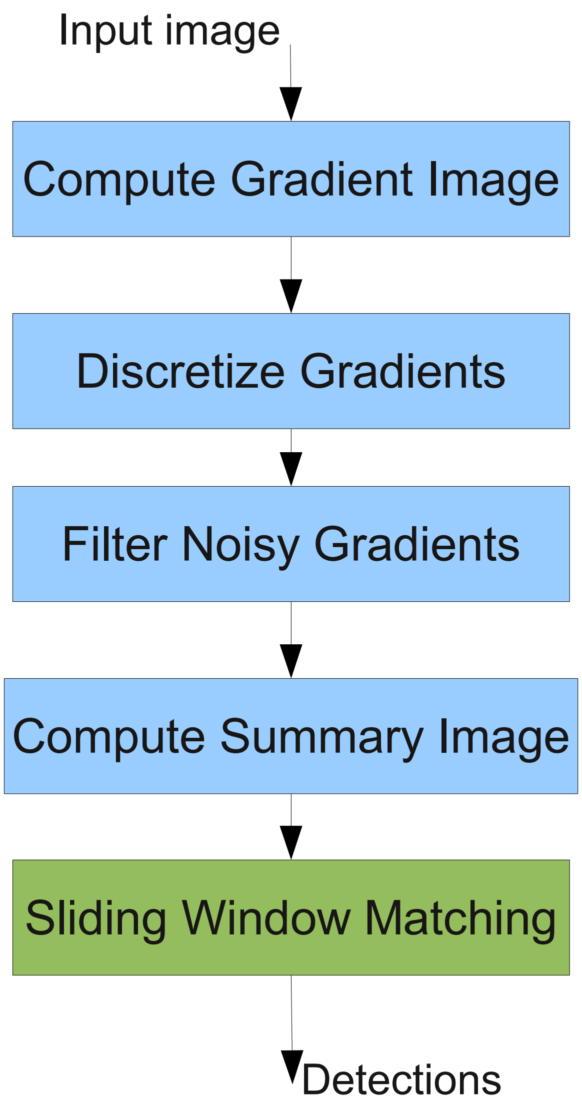
Willow Garage

BiGG – Algorithm Outline

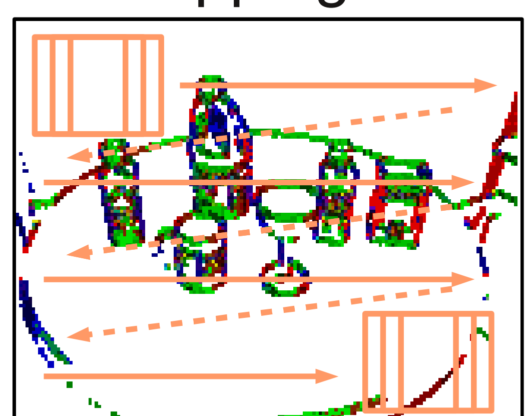


- Noisy gradient are filtered by non-maxima suppression on 3x3 cells
  - Discard singleton values (shot noise)
- A summary image is computed by down-sampling the discretized gradient image
  - Split the image in nxn cells (n=8)
  - OR the gradients in each cell
  - Speeds up the matching and makes it more robust to small shifts

BiGG – Algorithm Outline



- Slide a template over the image and compute response at each location
- The score is computed by an AND operation between the template and the image region
  - If above a threshold is considered a detection
- Apply non-maxima suppression to eliminate overlapping detections



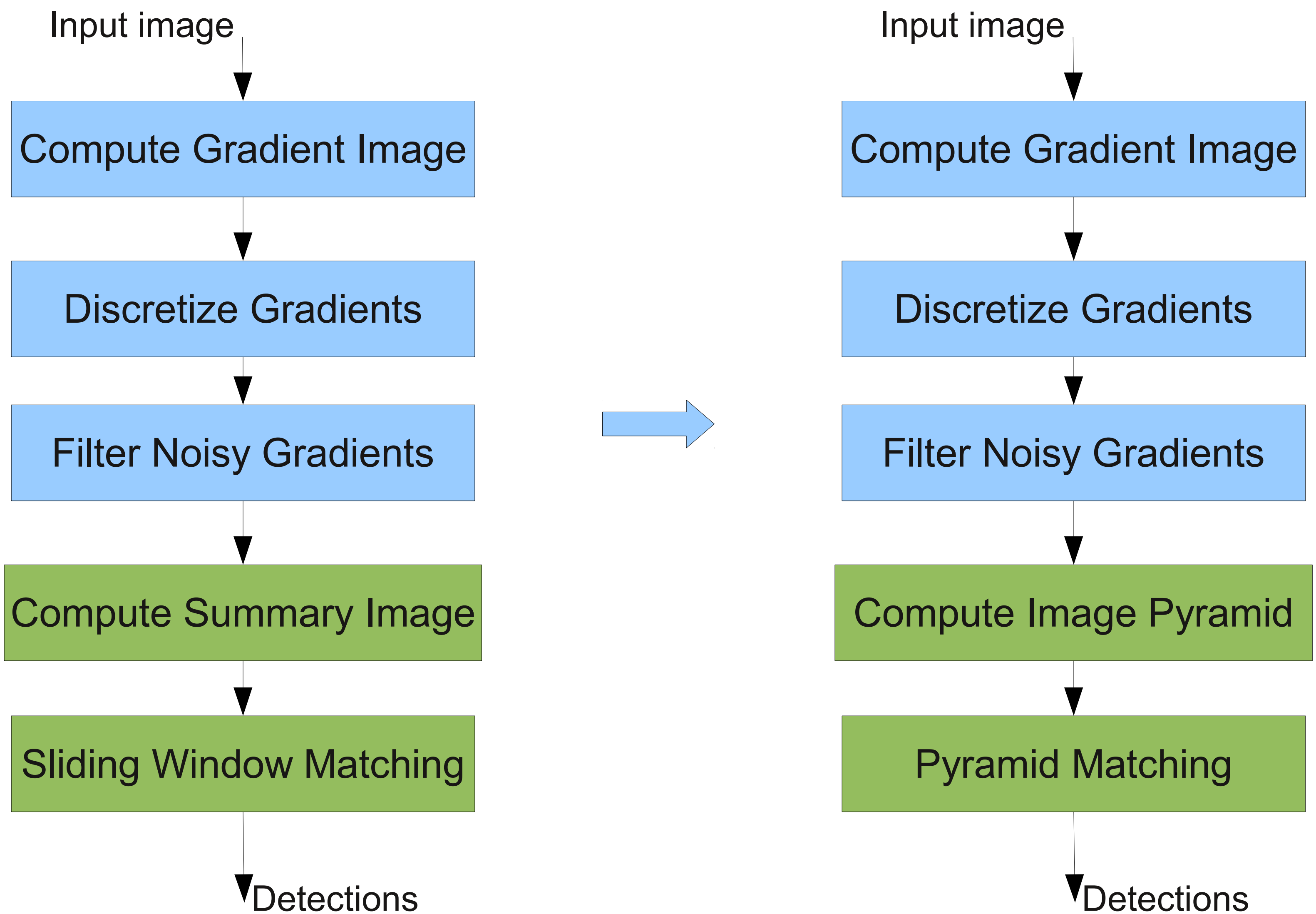
BiGG Limitations

- Sliding window approach
  - Large *image search space*
- Not scalable
  - Number of templates grows linearly with the number of objects
  - Large *template search space* (for a large number of templates)

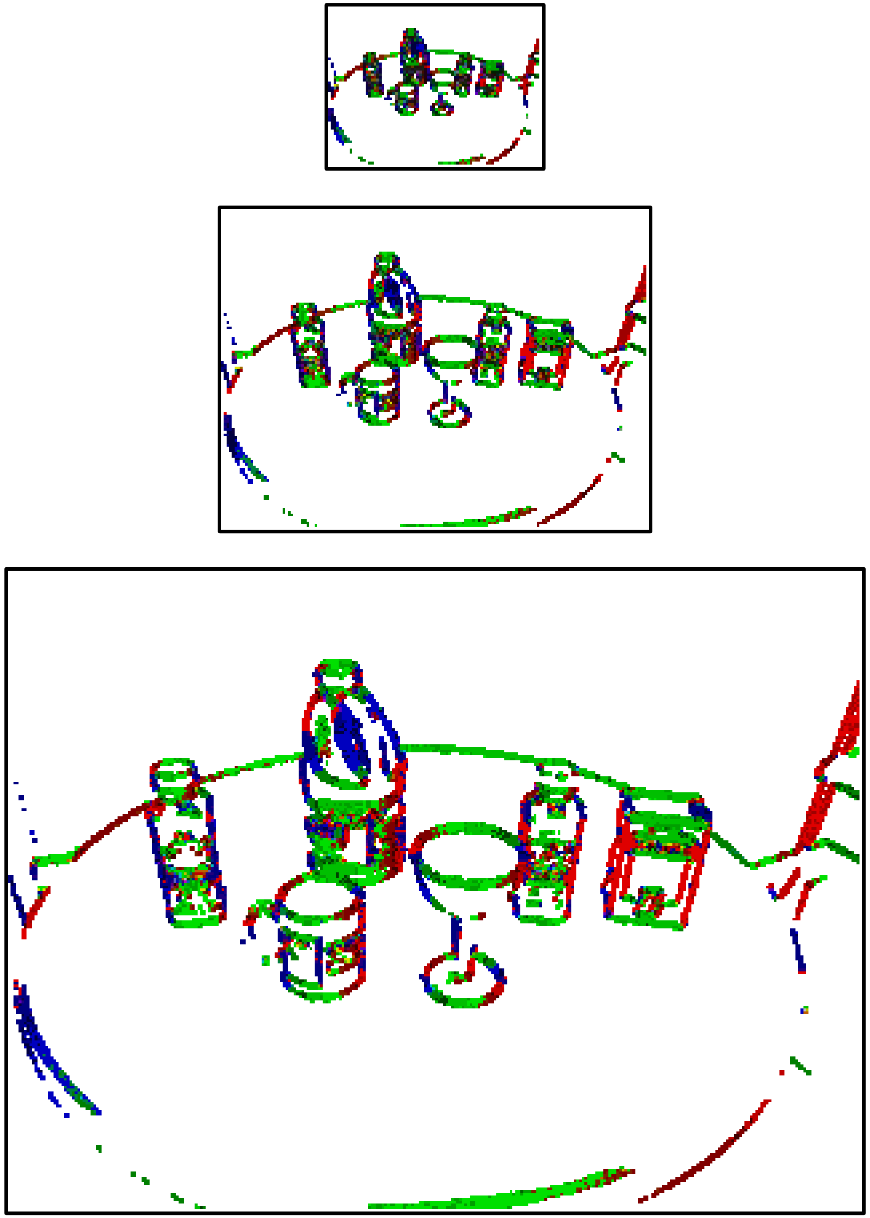
Scaling BiGG

- Binarized Gradient Grids Pyramid
  - Use a pyramid of binarized gradient images instead of a single down-sampled gradient image
  - Index the templates in a tree structure that mirrors the image pyramid
    - Small resolution templates on the root nodes, high resolution templates on leaf nodes
  - Reduces both the *image search space* and the *template search space*

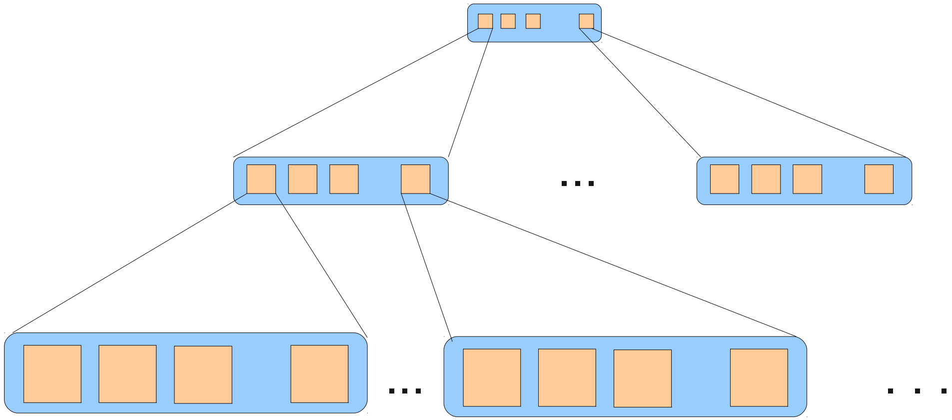
BiGGPy


Input image
Compute Gradient Image
Discretize Gradients
Filter Noisy Gradients
Compute Summary Image
Sliding Window Matching
Detections
Input image
Compute Gradient Image
Discretize Gradients
Filter Noisy Gradients
Compute Image Pyramid
Pyramid Matching
Detections

Computing Image Pyramid



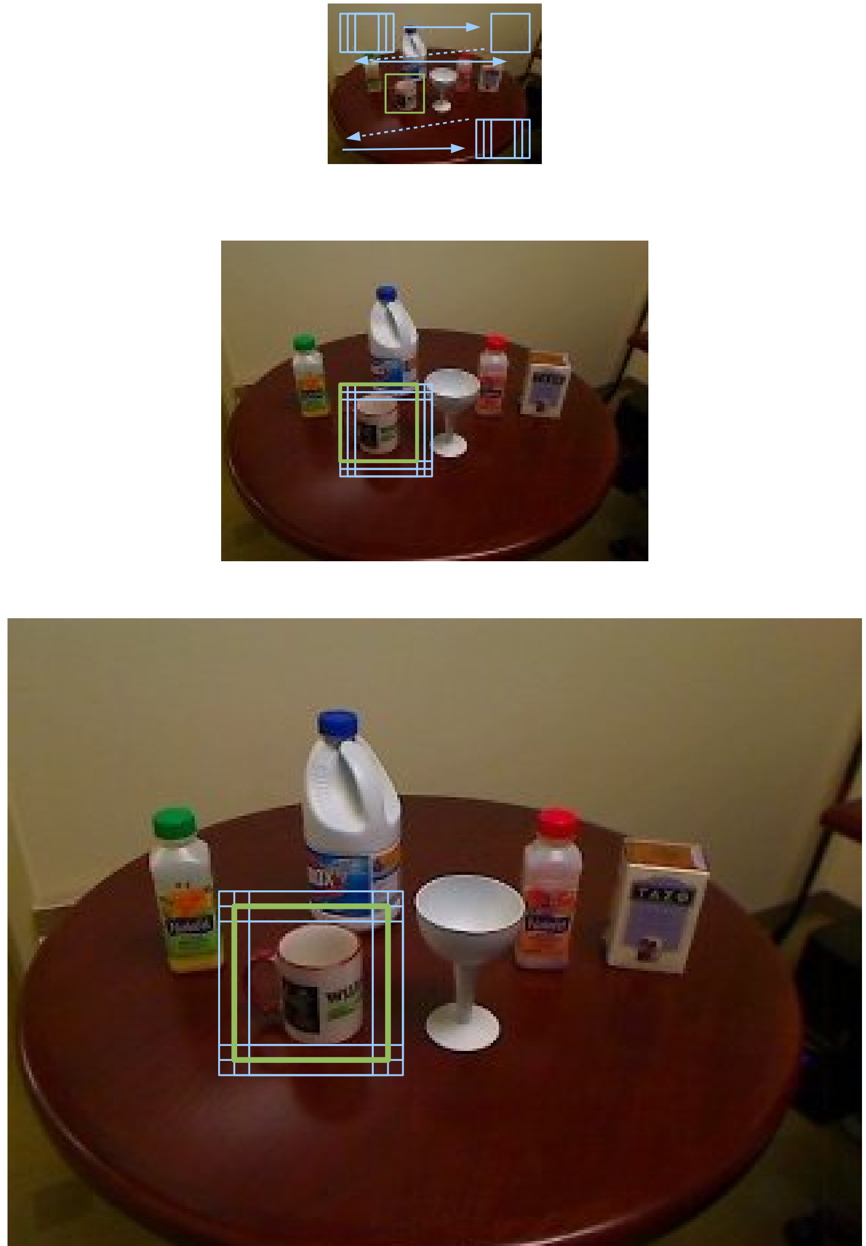
- Each level of the pyramid is computed by OR-ing together 2x2 cells from the lower level
- Templates are indexed in a tree that mirrors the structure of the image pyramid



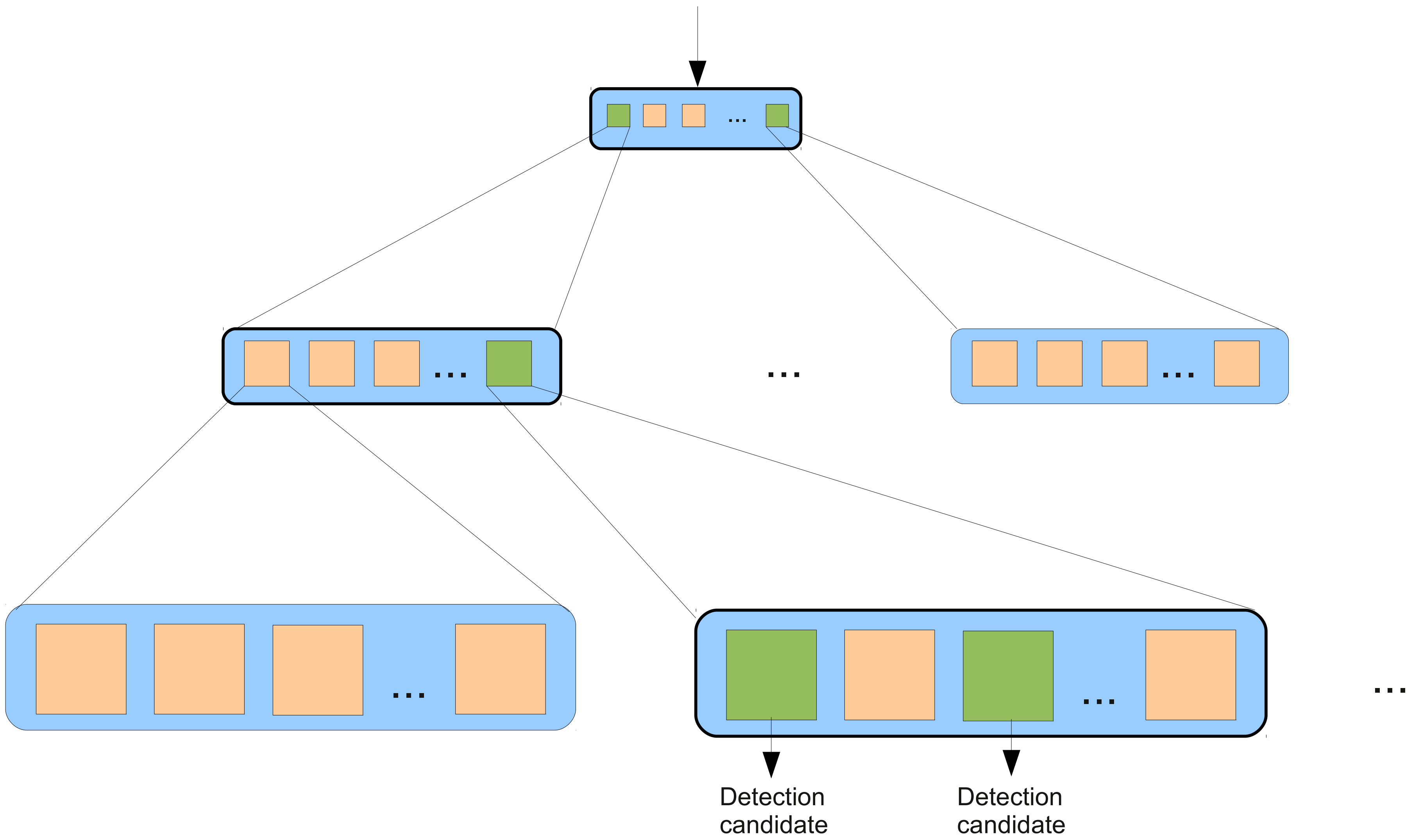
Pyramid Matching

- Start at top level with sliding window matching
  - Fast due to low resolution of the gradient image and few templates of low resolution
- For each of the detections on the top level search the next level in that neighborhood using the children templates of the template that matched at the top level
- Repeat previous step for all the levels
- Return detections on the lowest level

Image Search Space Reduction



Template Space Search Reduction

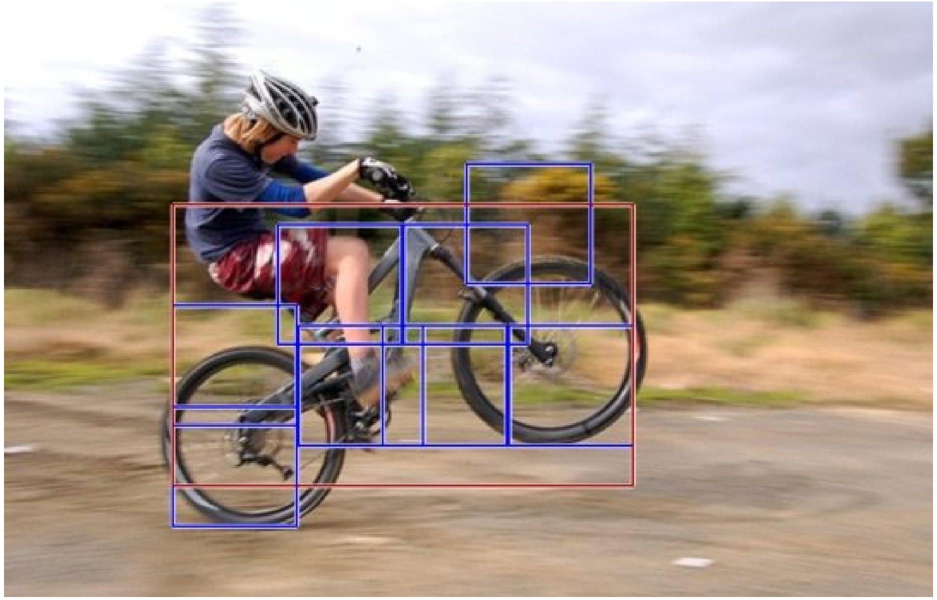


Demo

Demo...

Other work

- Deformable Part Models object detector
  (P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan)
  - One of the top performers in the VOC challenge
  - Wrapped it to work inside the recognition pipeline (`'dpm_detector'` package)
  - Scales linearly with the number of objects
  - High training time



Future Work

- Integrate BiGGPy with VFH (Viewpoint Cluster Histogram) Classifier (in progress)
  - VFH would filter out false positives and estimate pose of the object
- Do a quantitative evaluation on a large object dataset
  - Confirm the sub-linear scalability with respect to number of objects
- Use the compute cluster to scale to a very large number of objects

Thank you!