Lab Nov 8th

template matching

- Problem
  - Search over entire image
    - If image / template are large, slow
    - |template|*|image| pixel comparisons
  - Rotation invariance
  - Size invariance
  - many cases – very slow!!!

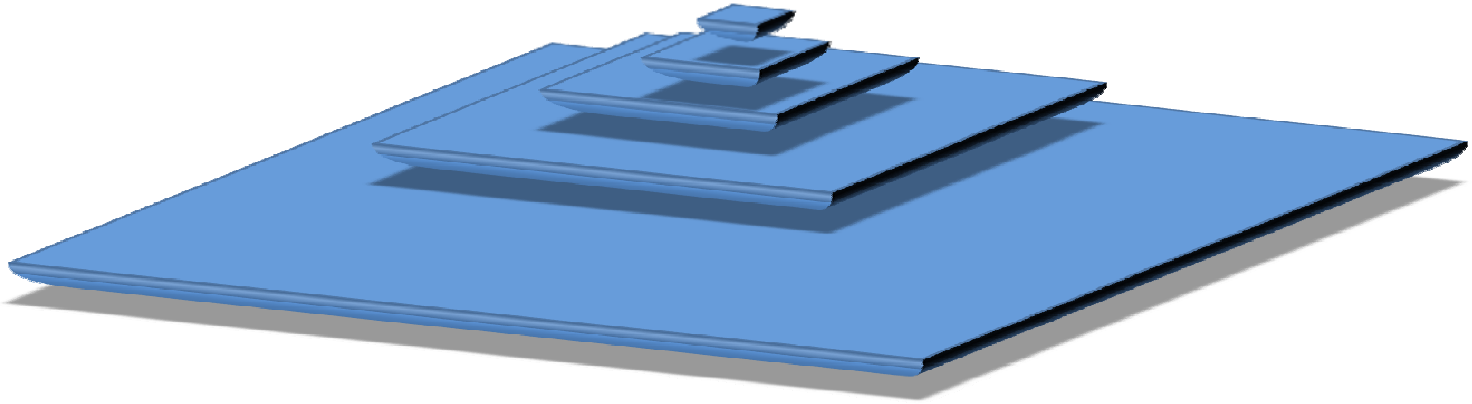
- also true for other image processing techniques

a look at complexity

- Ex 1
  - 1024x768 image = 786432 pixels 80x80 template = 6400 pixels
    - = >5 billion pixel comparisons
- Ex 2
  - 512x384 image = 196608 pixels 40x40 template = 1600 pixels
    - = 314 million pixel comparisons
- 256x192 image with 20x20 template = 19 million comparisons
- 128x96 with 10x10 = 1.2 million

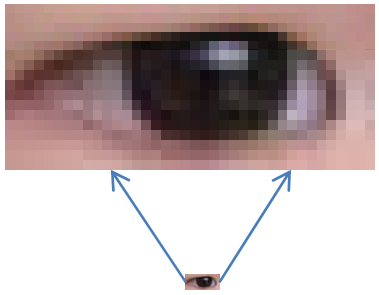
Each step has a factor of 16x speedup. Level1-4 = 4096X speedup

image pyramid



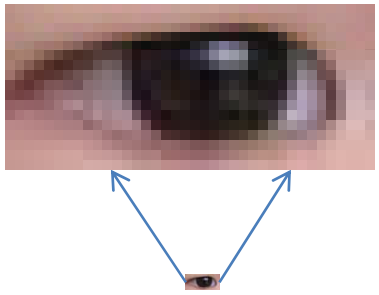
how can we use it?

- Many, many ways…
- size invariant-
  - same size template over image pyramid



how can we use it?

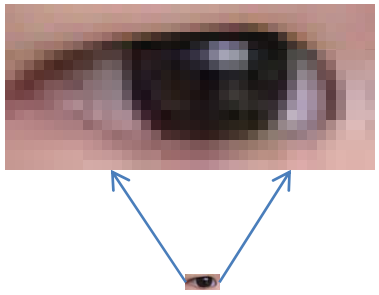
- Many, many ways…
- size invariant-
  - same size template over image pyramid



- Is this better/worse than resizing template?

how can we use it?

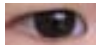
- Many, many ways…
- size invariant-
  - same size template over image pyramid

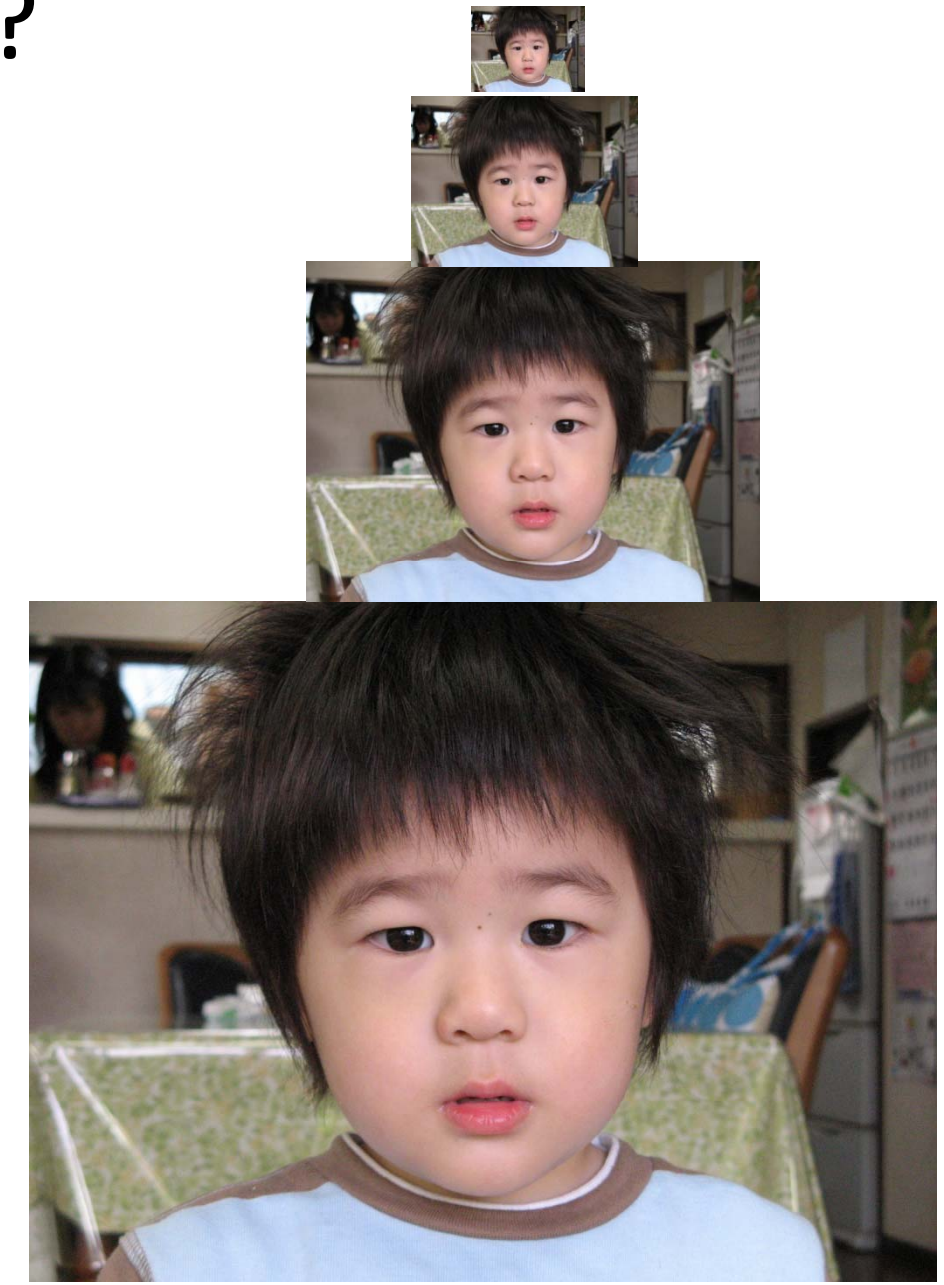


- Is this better/worse than resizing template? Better!

how can we use it?

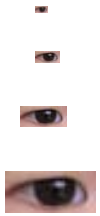
- prune search space

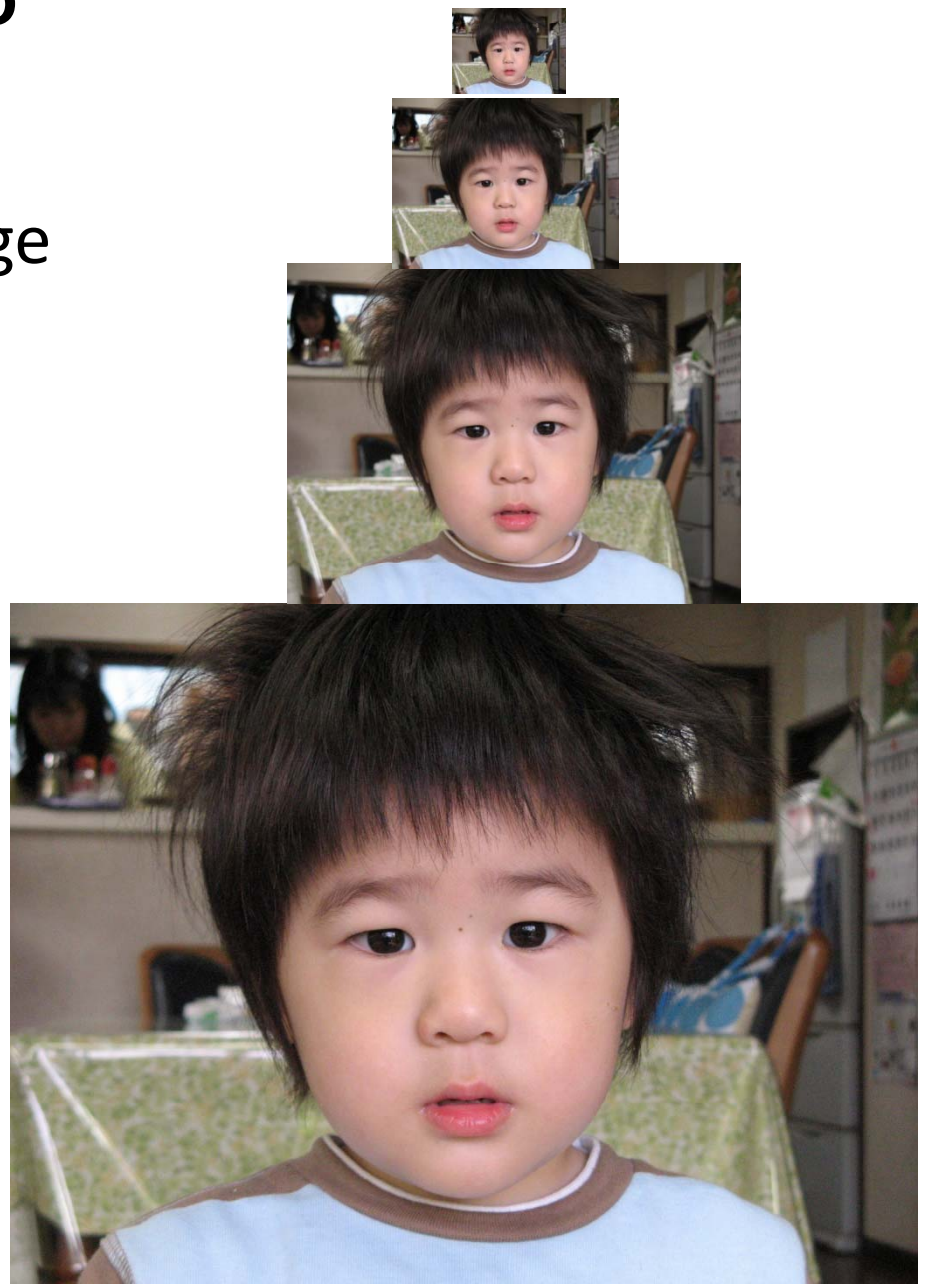




how can we use it?

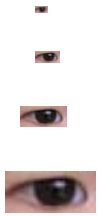
- prune search space
- 1. scale template with image

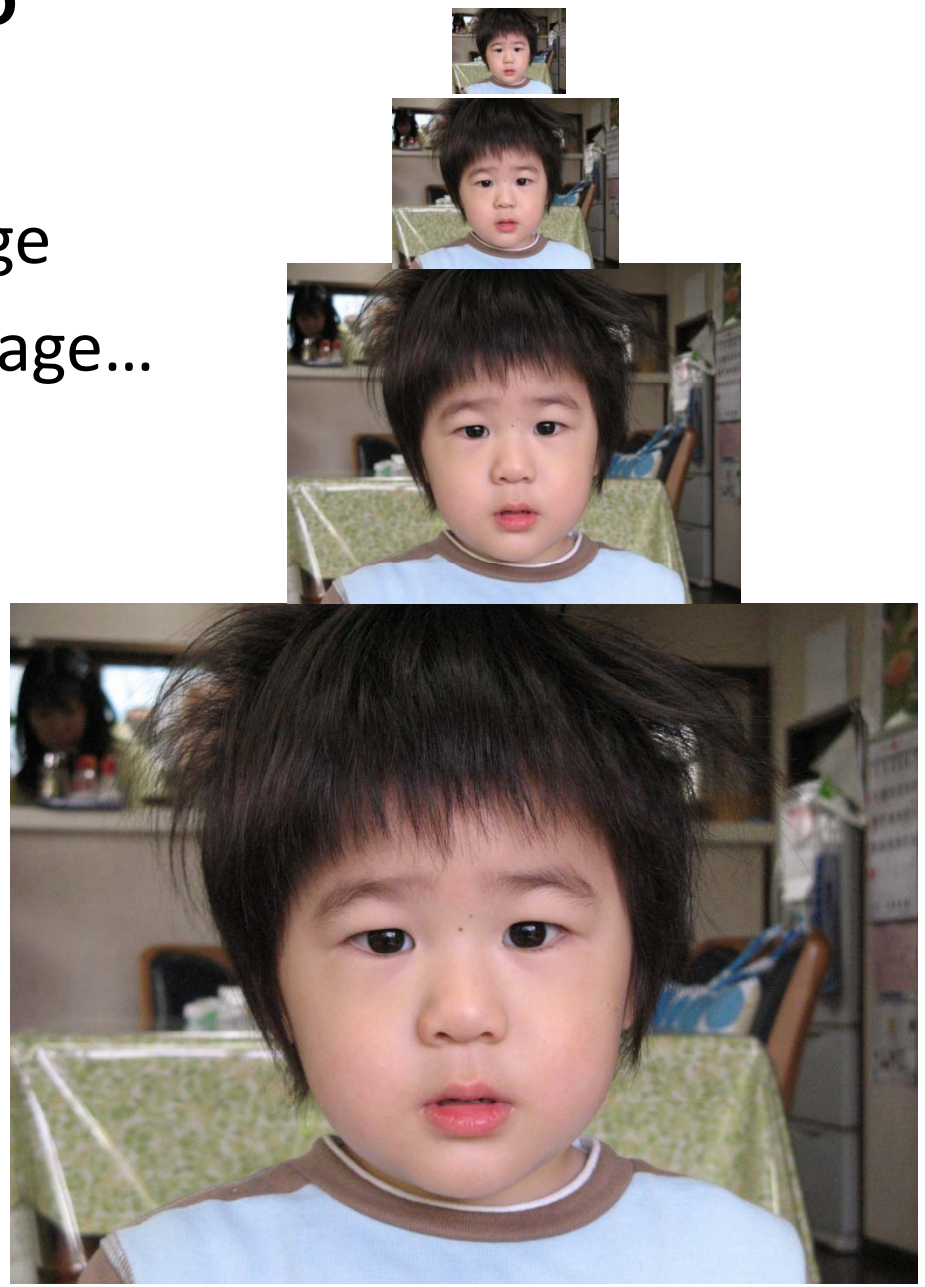




how can we use it?

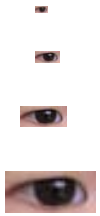
- prune search space
- 1. scale template with image
- Compare template with image…

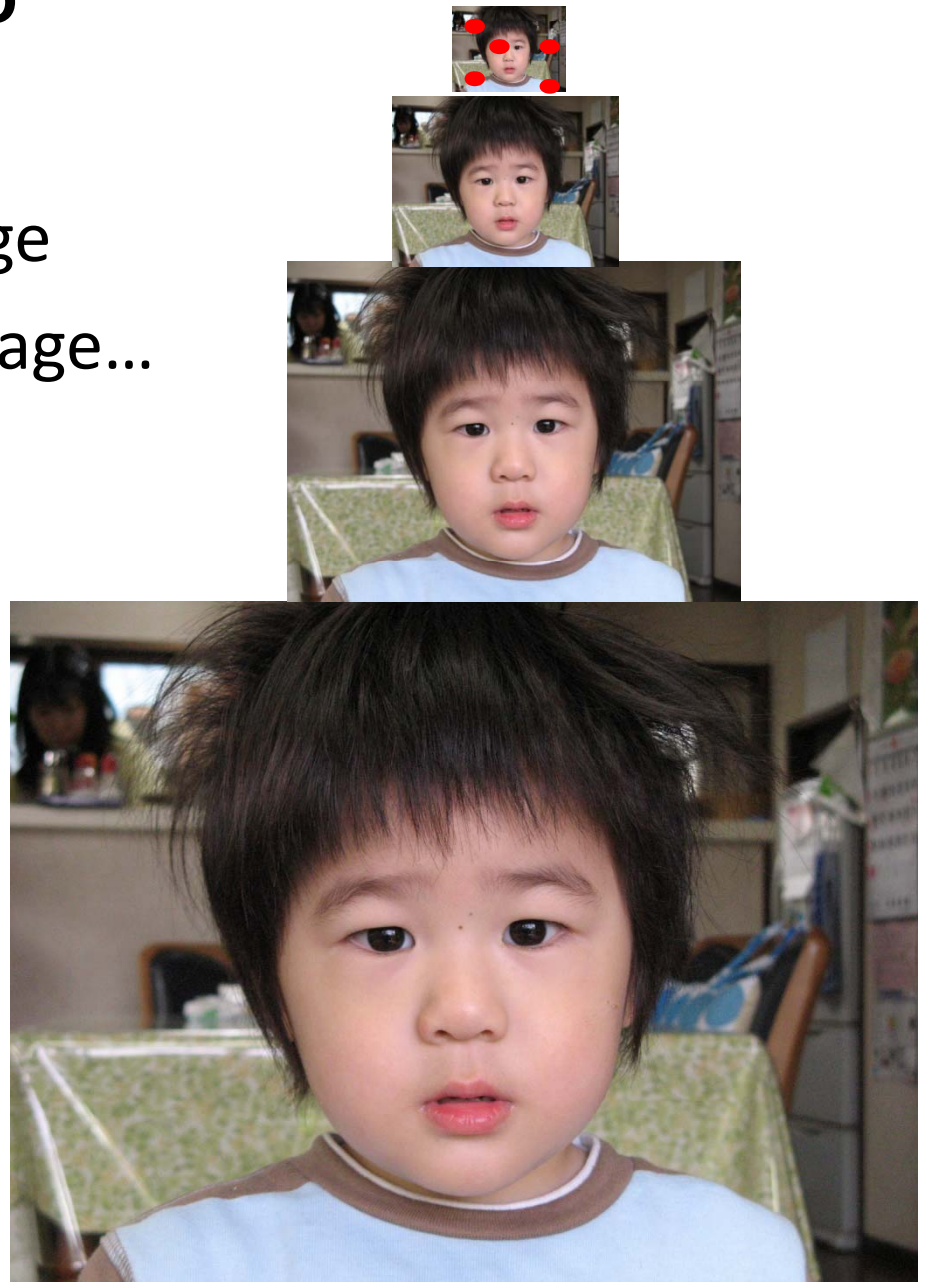




how can we use it?

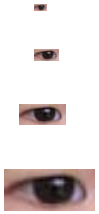
- prune search space
- 1. scale template with image
- Compare template with image…

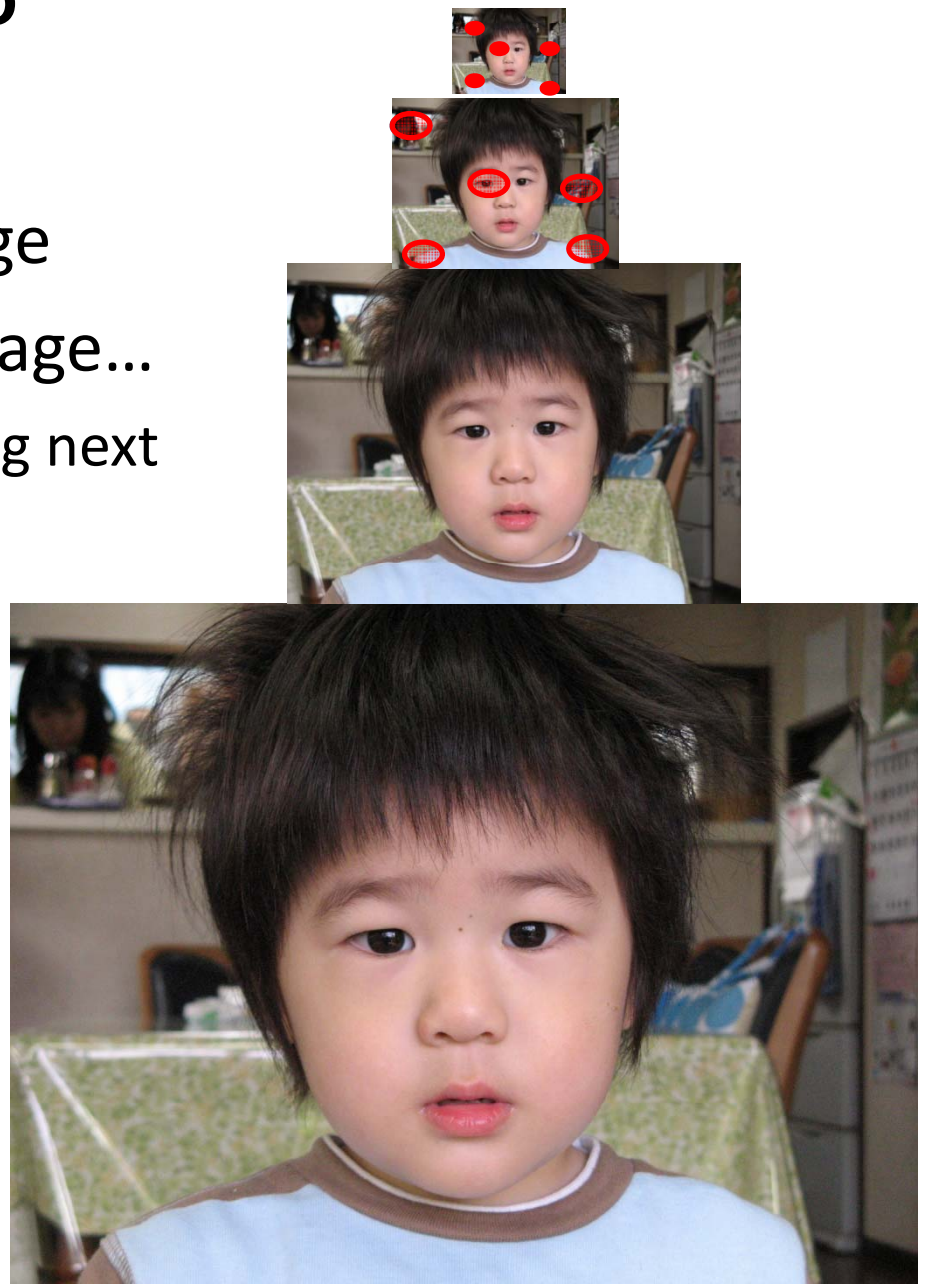




how can we use it?

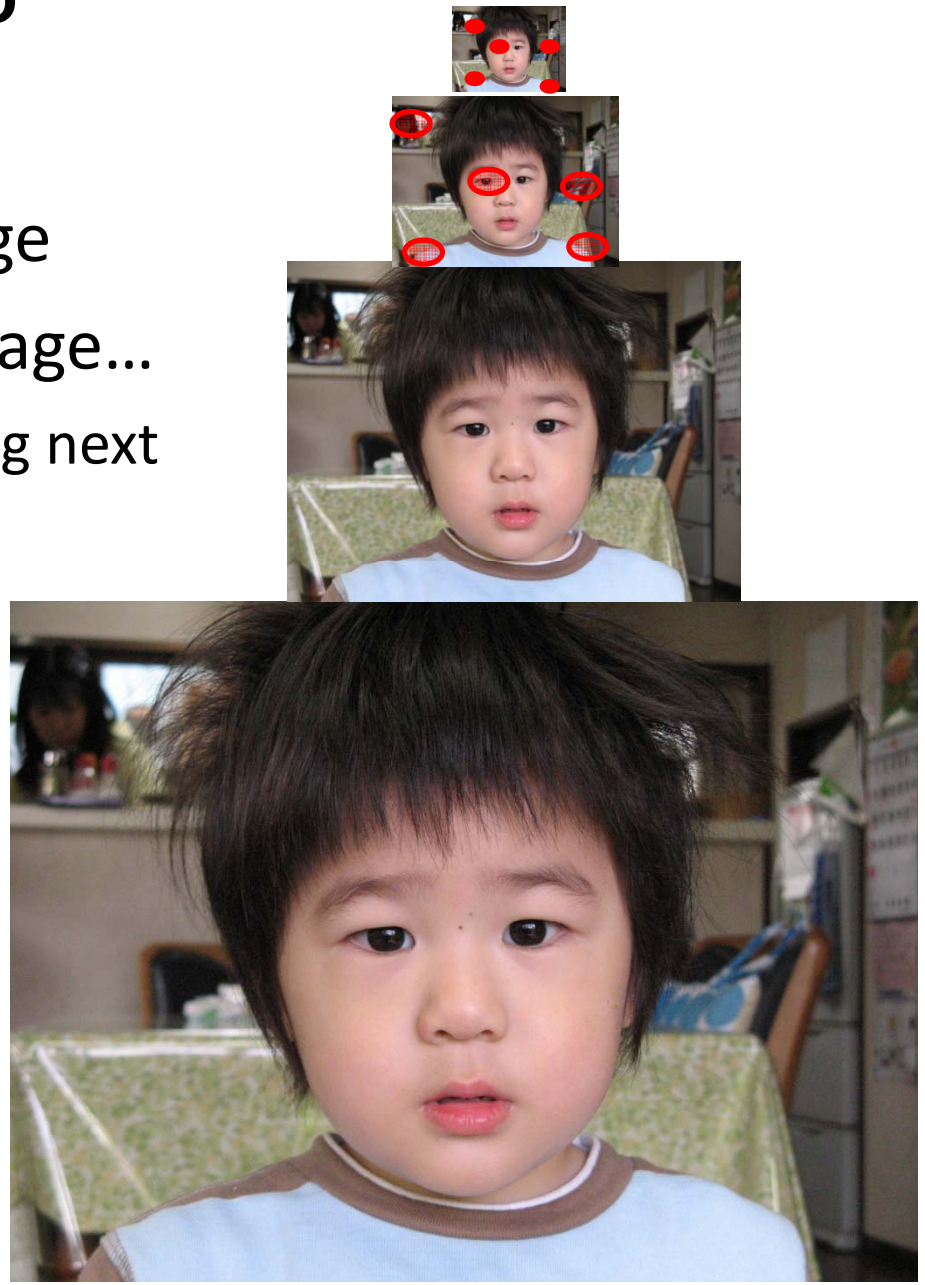
- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template





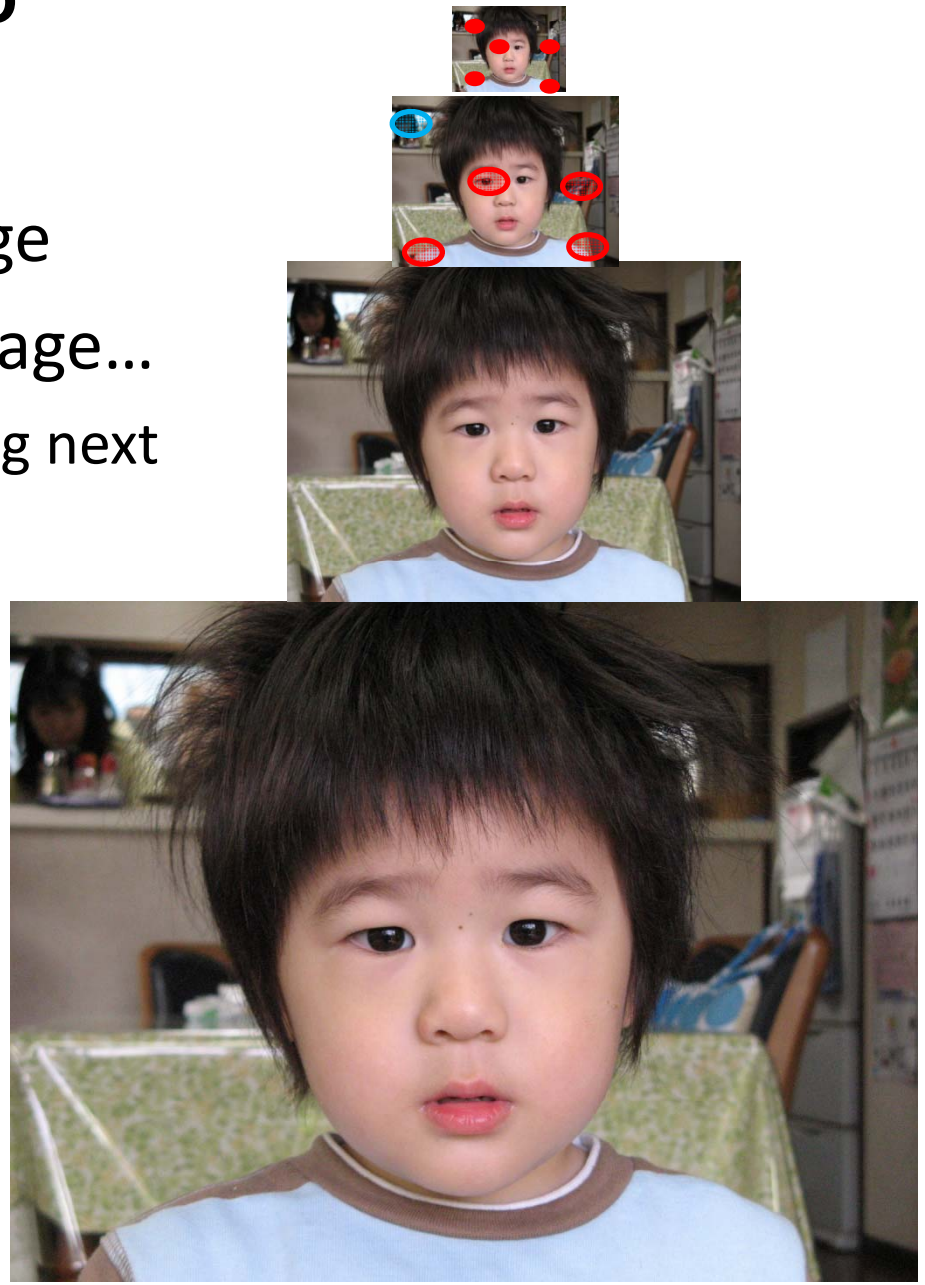
how can we use it?

- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template
  - repeat



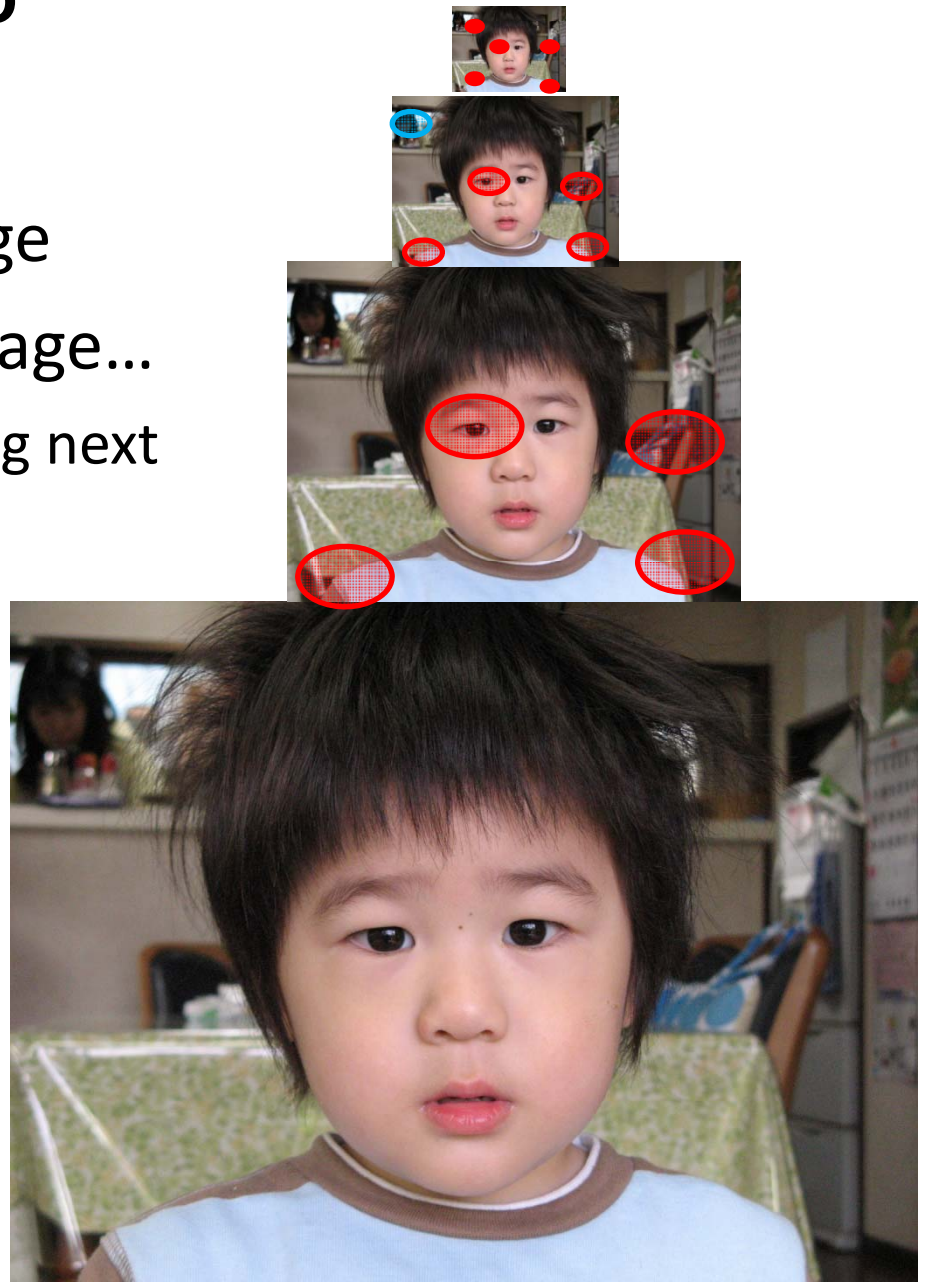
how can we use it?

- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template
  - repeat



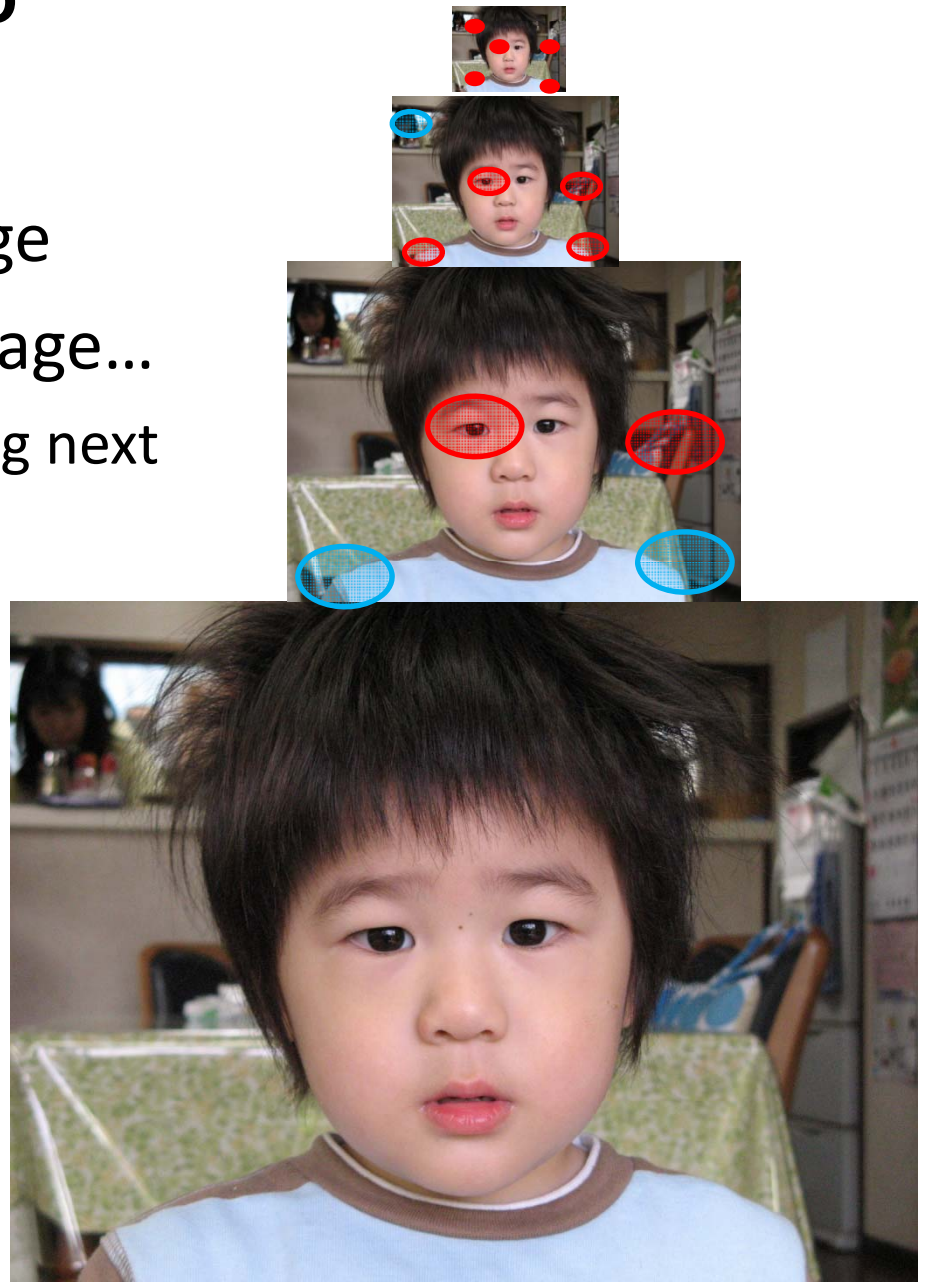
how can we use it?

- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template
  - repeat



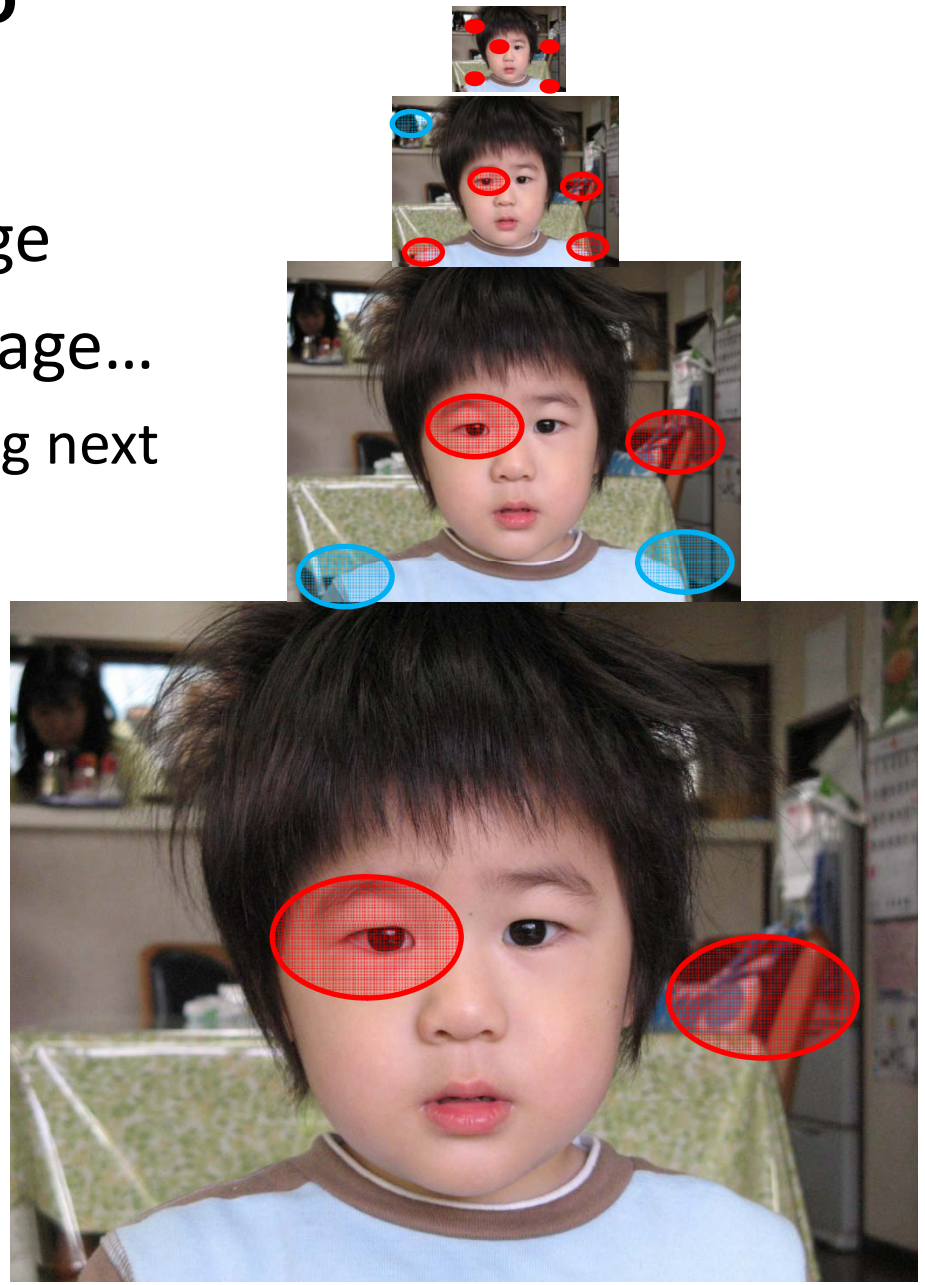
how can we use it?

- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template
  - repeat



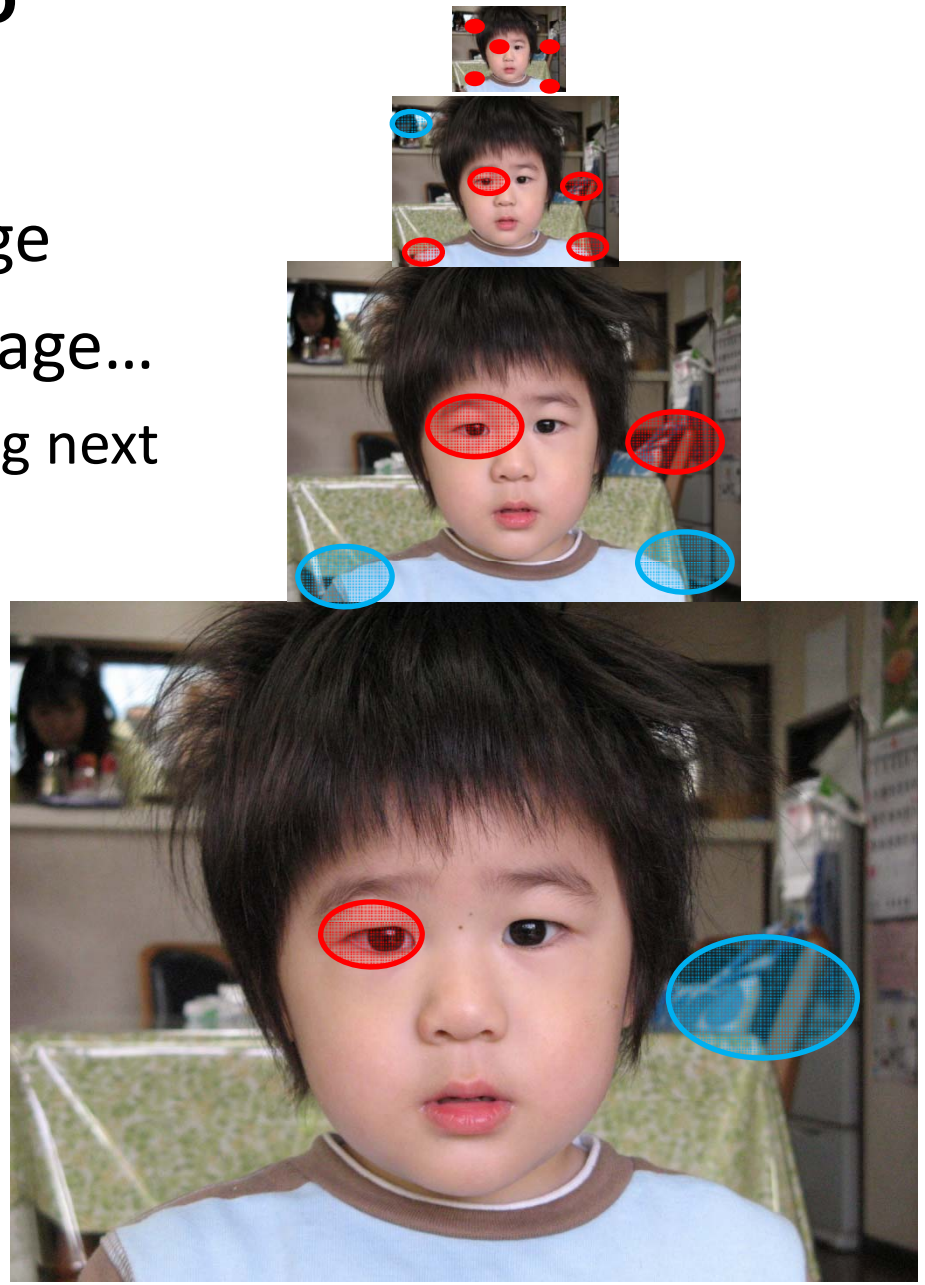
how can we use it?

- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template
  - repeat



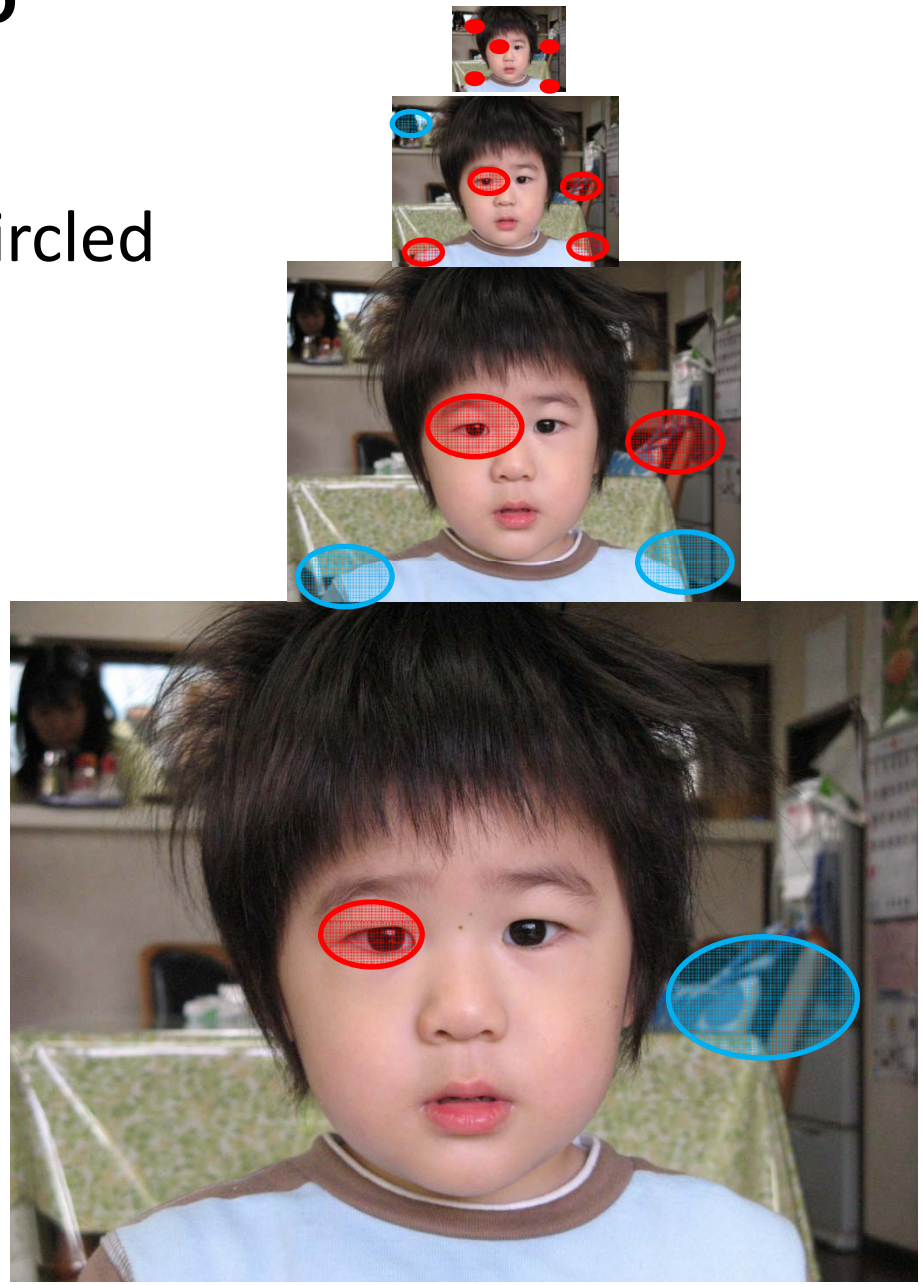
how can we use it?

- prune search space
- 1. scale template with image
- Compare template with image…
  - Look at subregions only using next template
  - repeat



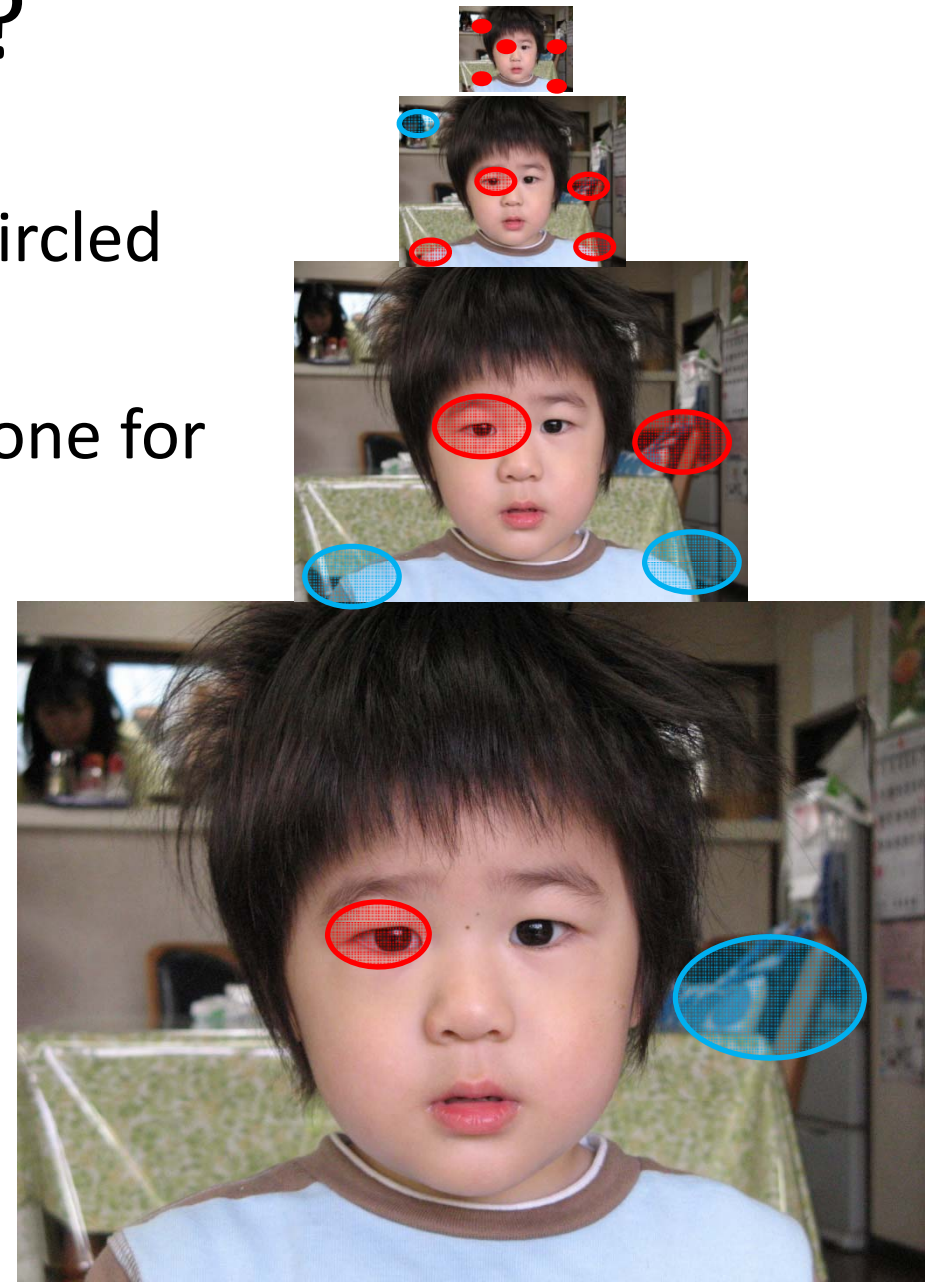
how can we use it?

- prune search space
- Only smallest image, and circled regions, were searched



how can we use it?

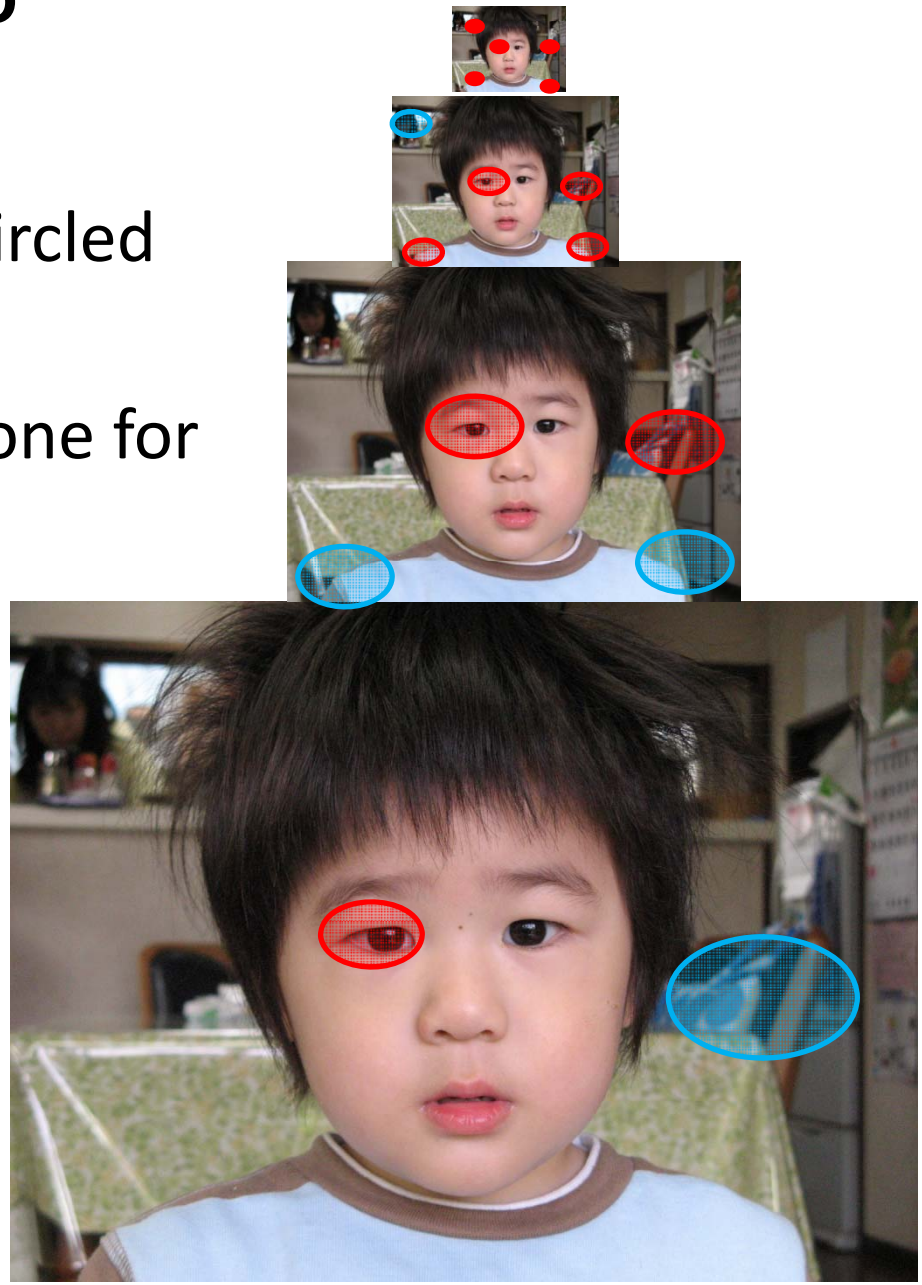
- prune search space
- Only smallest image, and circled regions, were searched
- similar technique can be done for any filter – low quality on low-res version, then high on limited regions



how can we use it?

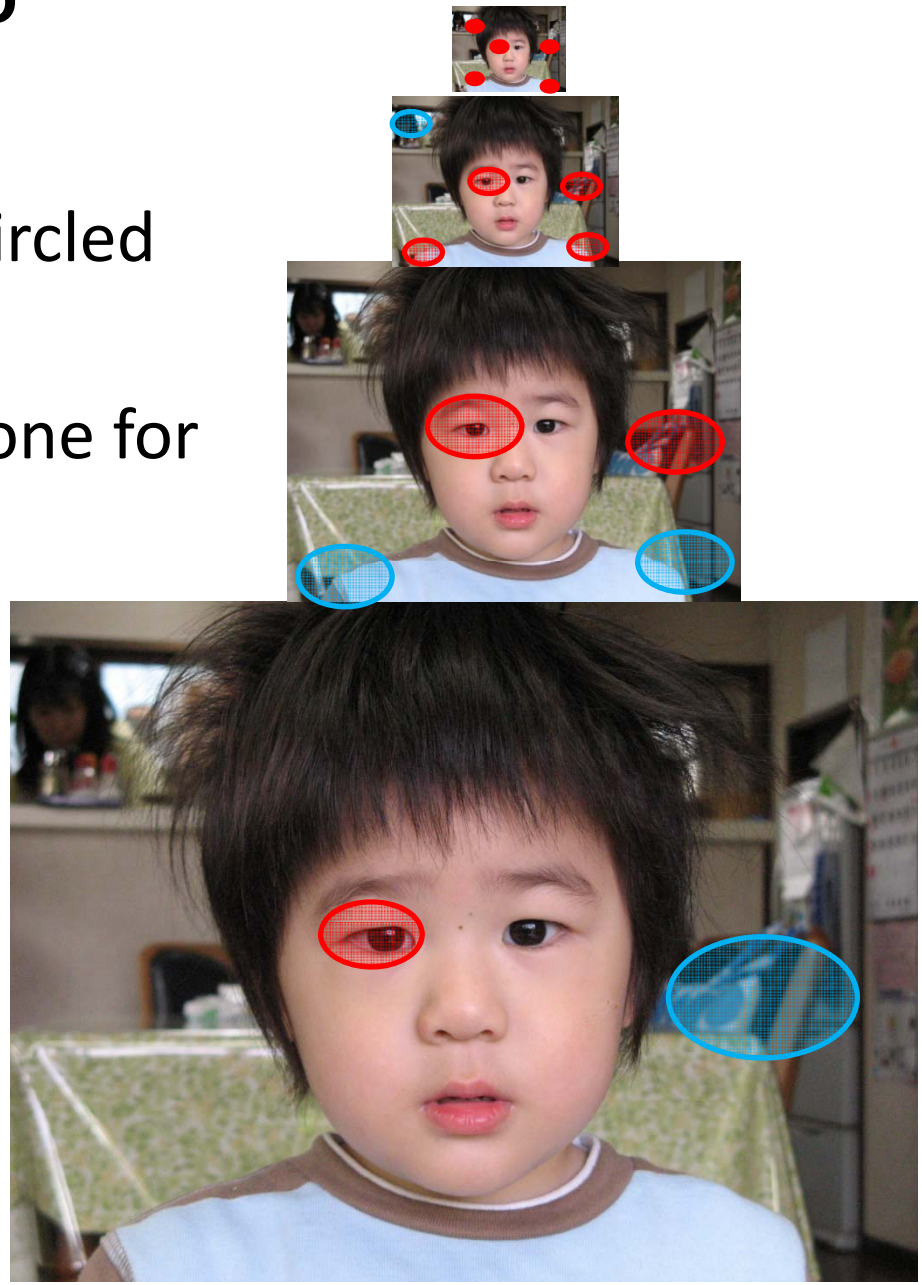
- prune search space
- Only smallest image, and circled regions, were searched
- similar technique can be done for any filter – low quality on low-res version, then high on limited regions

- What kinds of problems does this introduce?



how can we use it?

- prune search space
- Only smallest image, and circled regions, were searched
- similar technique can be done for any filter – low quality on low-res version, then high on limited regions



- What kinds of problems does this introduce?
  - Miss at early level could ruin the detection

implementation
how to make pyramid

- Just take every 2^L pixel..
- [0 1 4 6 7 2 5 1 2 1 1]
- [0 4 7 5 2 1]
- [0 7 2]
- [0 2]

implementation
how to make pyramid

- or every 3^L pixel.. Or X^L
- [0 1 4 6 7 2 5 1 2 1 1]
- [0 6 5 1]
- [0 1]

- No extra memory required. Very fast

implementation
how to make pyramid

- or every 3^L pixel.. Or X^L
- [0 1 4 6 7 2 5 1 2 1 1]
- [0 6 5 1]
- [0 1]

- No extra memory required. Very fast
- Can easily lose features
  - (nearest neighbour problems)

implementation
how to make pyramid

- Generate pixel by smoothing (gaussian)
- [0 1 4 6 7 2 5 1 2 1 1]
- [0 4 7 5 2 1]
- [0 7 2]
- [0 2]