A Brief Primer on Inverse Procedural Modeling

CS334

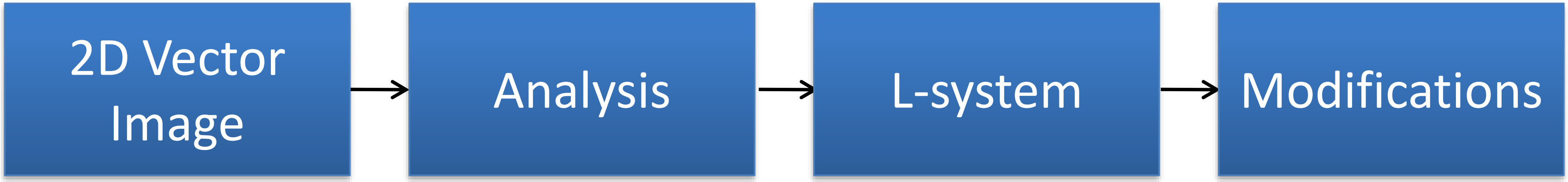
Daniel G. Aliaga
Department of Computer Science
Purdue University

Recall: Procedural Modeling

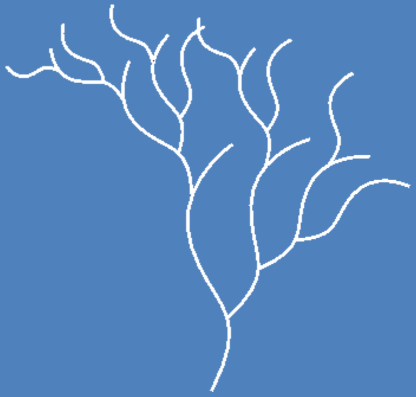
- Apply algorithms for producing objects and scenes
- The rules may either be embedded into the algorithm, configurable by parameters, or externally provided

#1: Inverse Procedural Modeling by Automatic Generation of L-systems

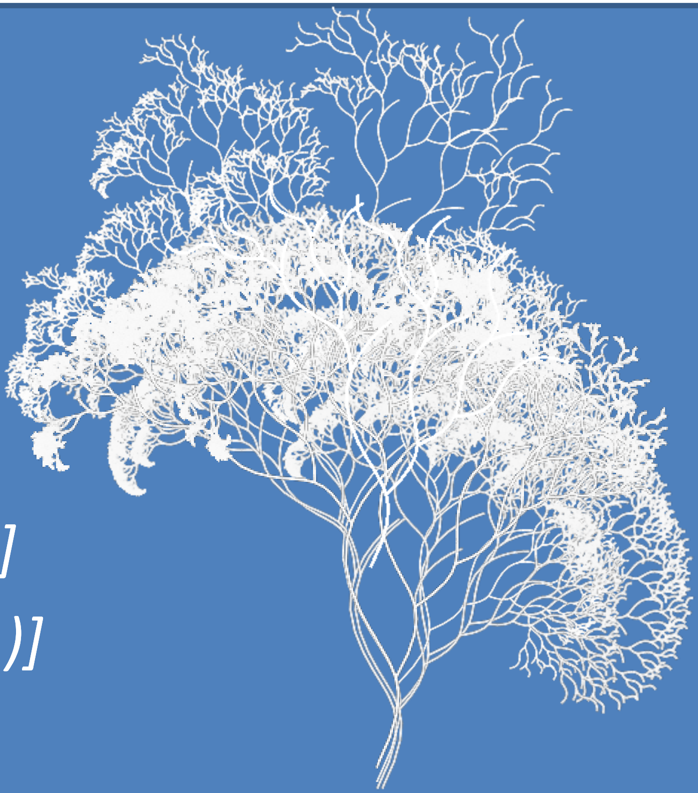
O. Šťava, B. Beneš, R. Měch, D. Aliaga, P. Krištof*



2D Vector Image → Analysis → L-system → Modifications



$$R(m) \rightarrow A[-(\alpha)\, f(d) * (s) - (\beta) R(m-1)][+(\alpha')\, f(d') * (s') + (\beta') R(m-1)]$$

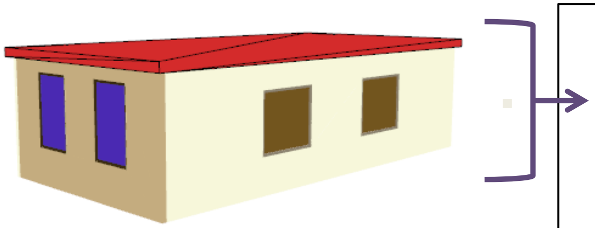


#2: Inverse 3D Procedural Buildings

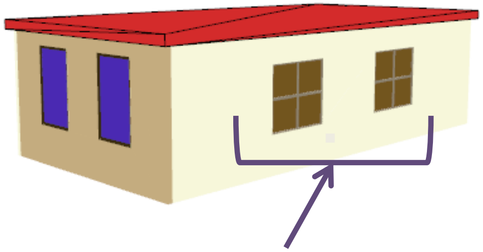
I. Demir, B. Benes, D. Aliaga

Triangle Soup

Grammar

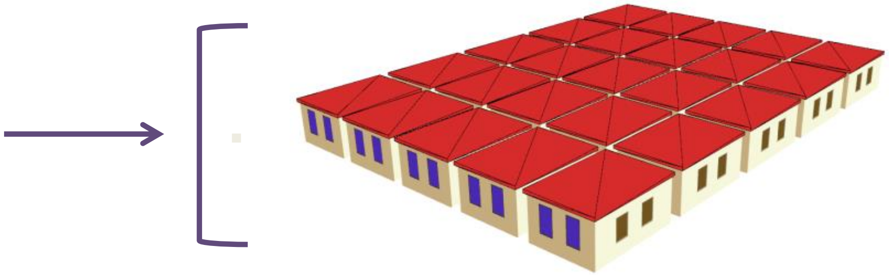


```
Rule Building(S) = {                        // CREATE ENTIRE BUILDING
    a = split(S, 0.0, 0.0, 0.80, 4) // roof terminal
    c = split(S, 1.0, 1.0, 0.01, 5) // floor terminal
    B = split(S, 1.0, 1.0, 0.80, 5) // wall bbox
    D = split(B, 1.0, 0.1, 1.00, 5) // front wall bbox
    E = split(B, 0.0, 0.9, 0.00, 4) // back wall bbox
    F = split(B, 0.1, 1.0, 1.00, 5) // left wall bbox
    G = split(B, 0.9, 0.0, 0.00, 4) // right wall bbox
    R3(D); R3(E);                   // front/back rule
    R4(F); R4(G);  }                // left/right rule
}
Rule R3(A) = {                              // CREATE FRONT/BACK WALL
    b = split(A, 0.0, 0.0, 0.00, 4) // wall terminal
    C = split(A, 0.4, 1.0, 0.75, 5)
    D = split(C, 0.6, 0.3, 0.00, 4) // left window bbox_
    E = split(A, 0.6, 1.0, 0.75, 6)
    F = split(E, 0.3, 1.0, 0.30, 1) // right window
    R1(D); R1(F);                   // window rule
}
Rule R1(A) = {
    // perform splits for frame and window
}
[...and add’l text for R2 and R4]
```

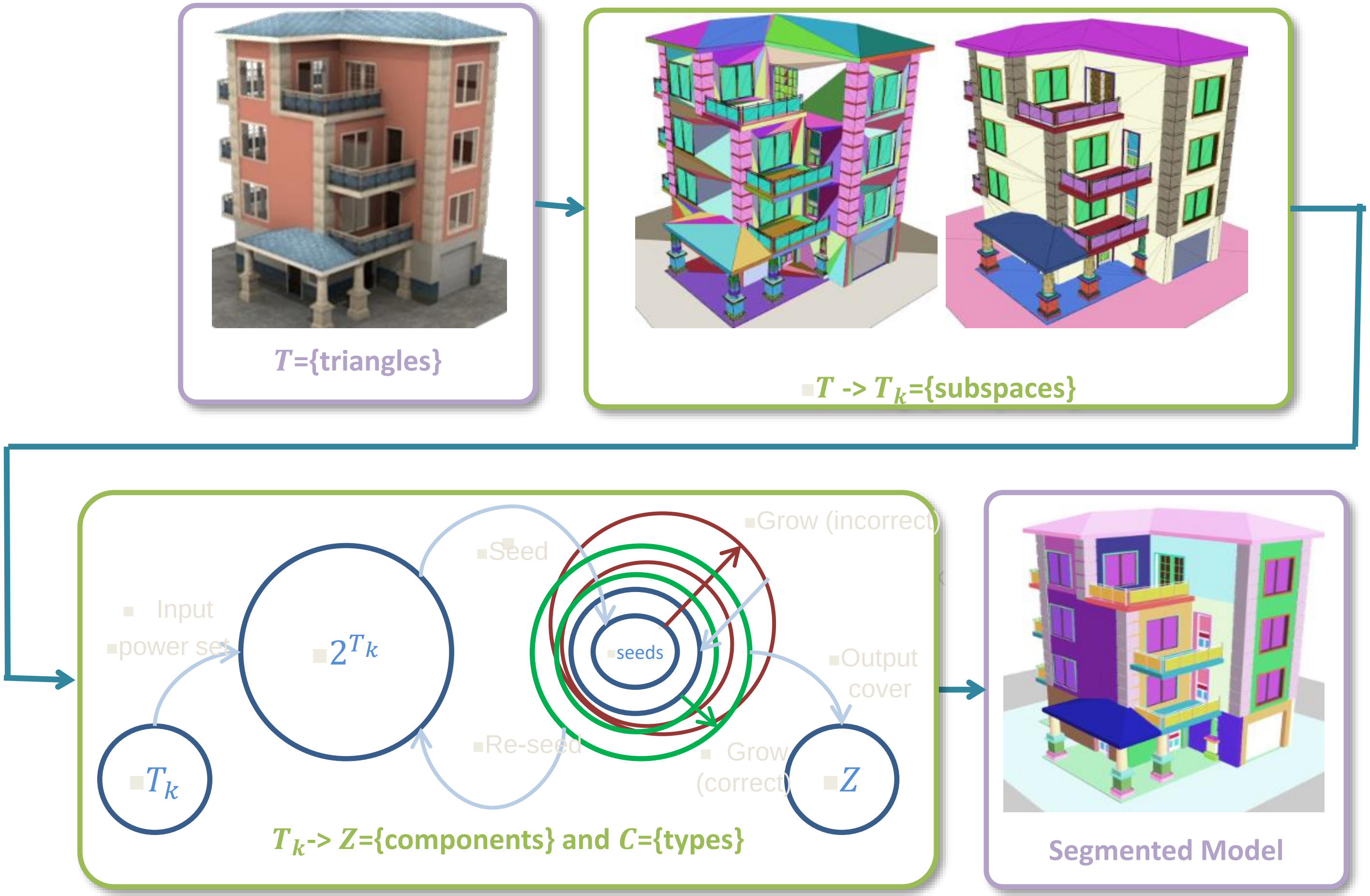


```
Rule R3(A) = {
    ...
    R1(subdivide(2,1,2,D));
    R1(subdivide(2,1,2,F));
}
```

```
Rule NewBuilding(S) = {
    Building(subdivide(5,5,1,S));
}
Rule Building(S) = {
    Z = split(S, 0.9, 0.9, 1.0, 5) // inset
    a = split(Z, 0.0, 0.0, 0.8, 4) // same
    ...
}
```



Pipeline



T={triangles}

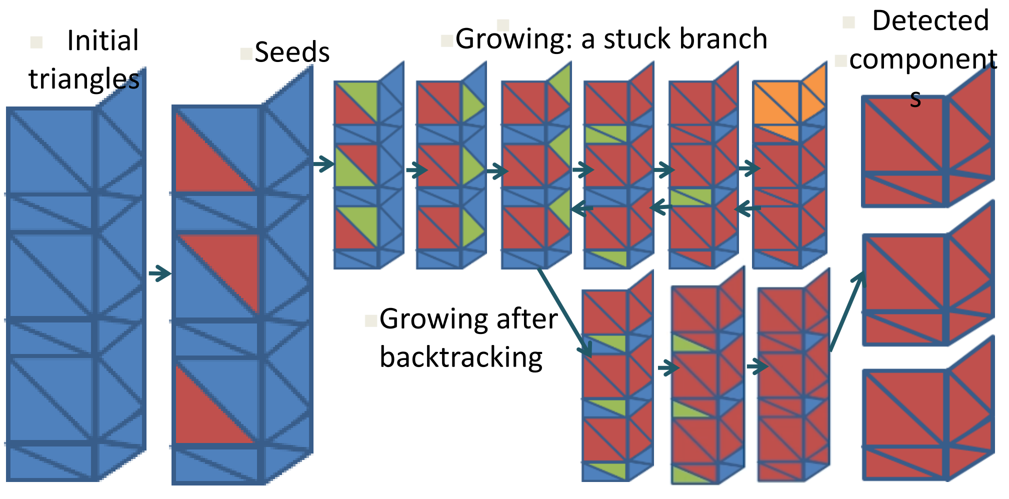
T -> T_k={subspaces}

T_k-> Z={components} and C={types}

Segmented Model

[Demir et al. 2015]

Search Process



Initial triangles

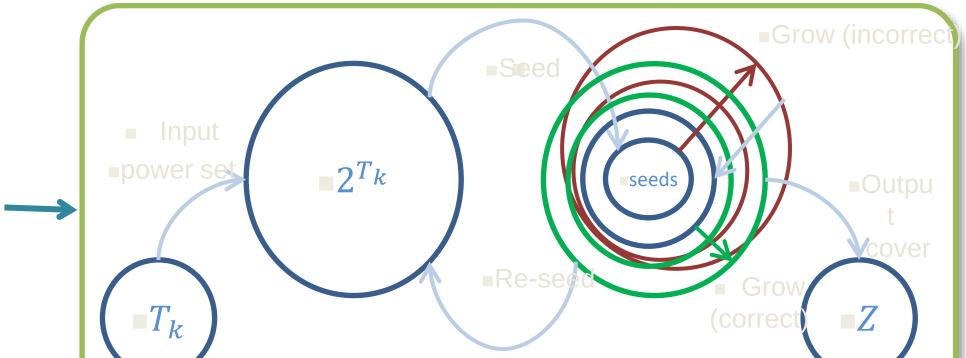
Seeds

Growing: a stuck branch

Detected component s

Growing after backtracking

Z-> Combinatorial Optimization -> M={components} and C={types}



Input power set

2^{T_k}

Seed

Grow (incorrect)

seeds

Output cover

Re-seed

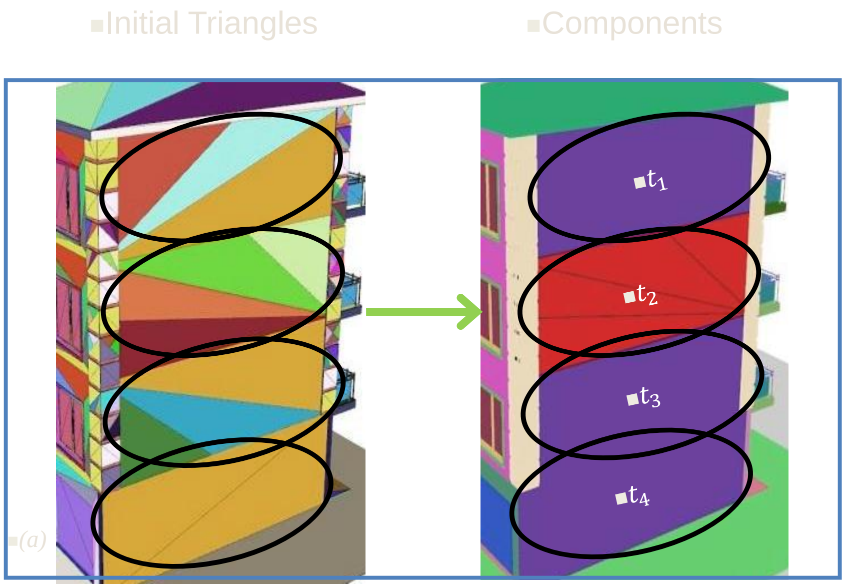
Grow (correct)

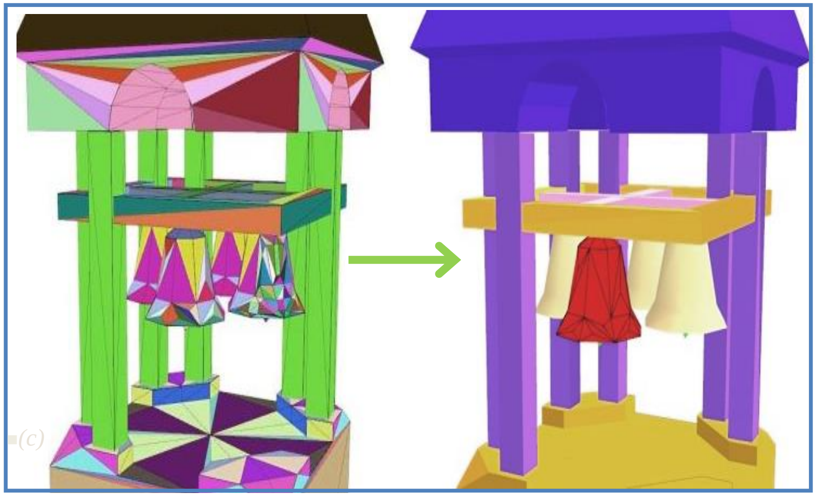
T_k

Z

T_k-> Z={components} and C={types}

Clustering and Components

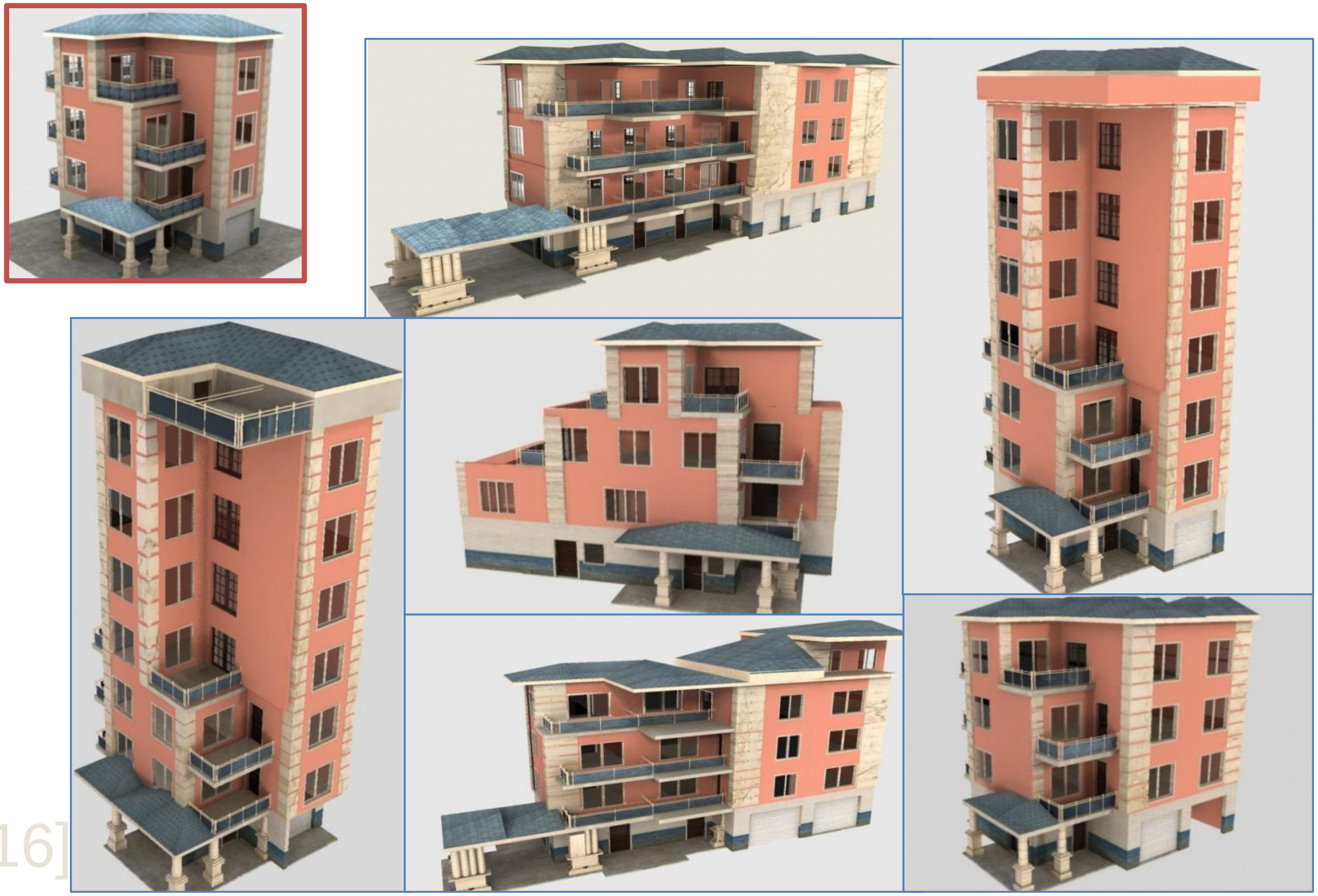




1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Example Output

- Automatic



[Demir et al. 2016]

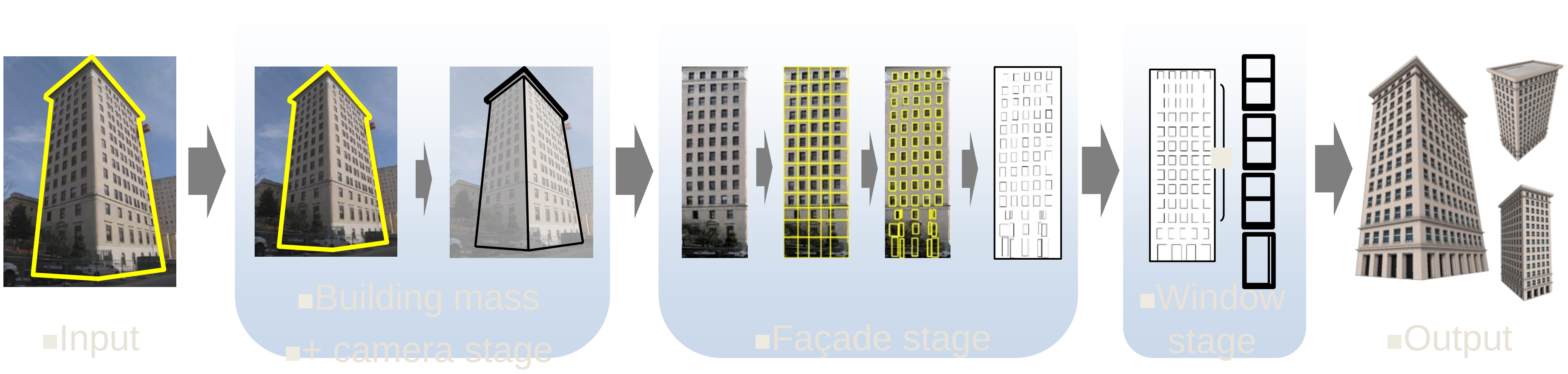
#3: Photograph to Proc. Model

G. Nishida, A. Bousseau, D. Aliaga

- Given an image of a building and its contour, generate a 3D procedural model similar to the input.

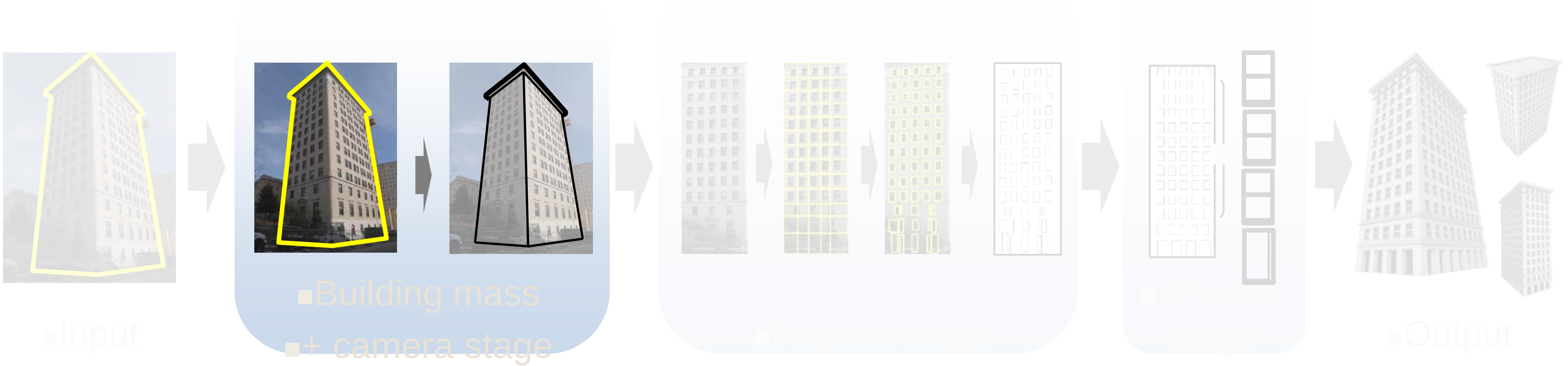
▪[Nishida et al.

System Pipeline



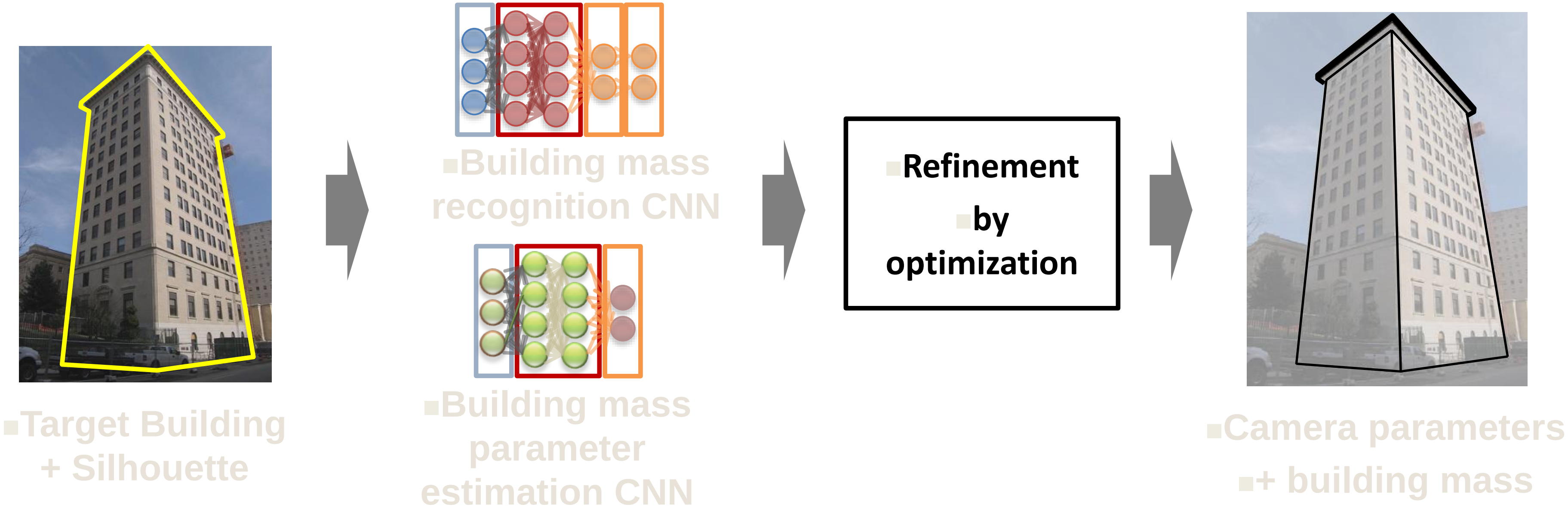
- Input
- Building mass
- + camera stage
- Façade stage
- Window stage
- Output

Building Mass + Camera Stage



- Input
- Building mass
- + camera stage
- Façade stage
- Window stage
- Output

Building Mass + Camera Stage



Target Building + Silhouette

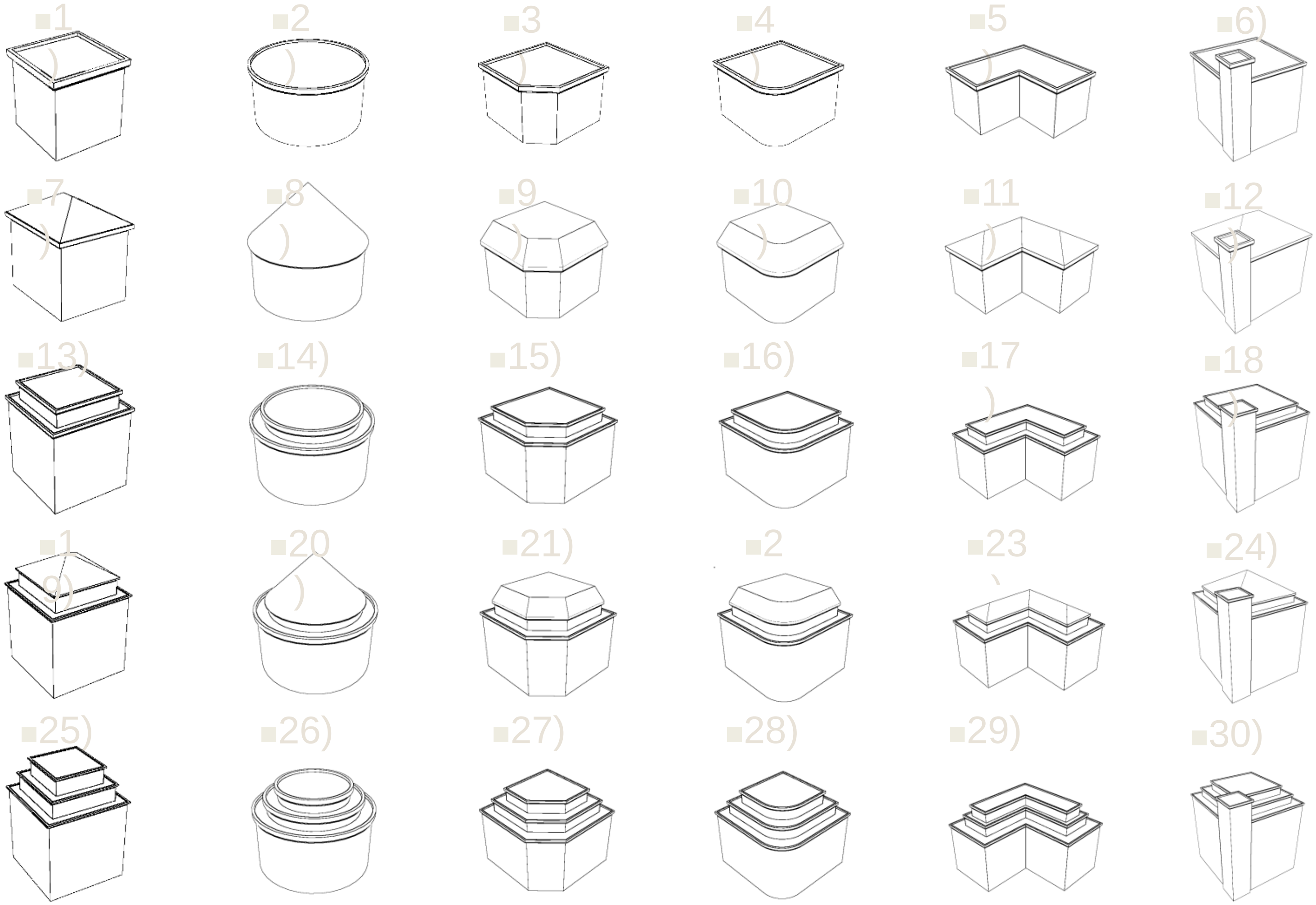
Building mass recognition CNN

Building mass parameter estimation CNN

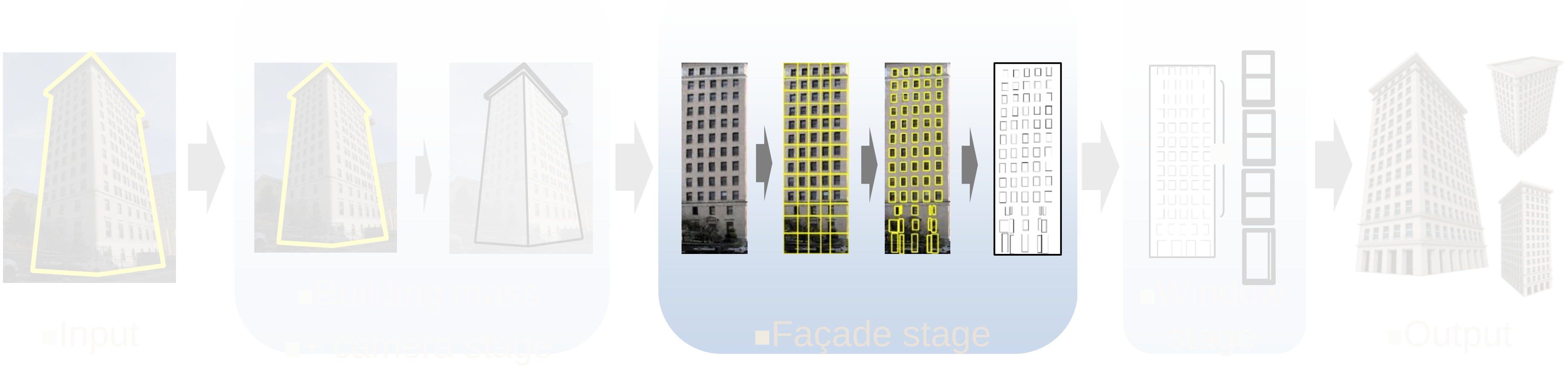
Refinement by optimization

Camera parameters + building mass

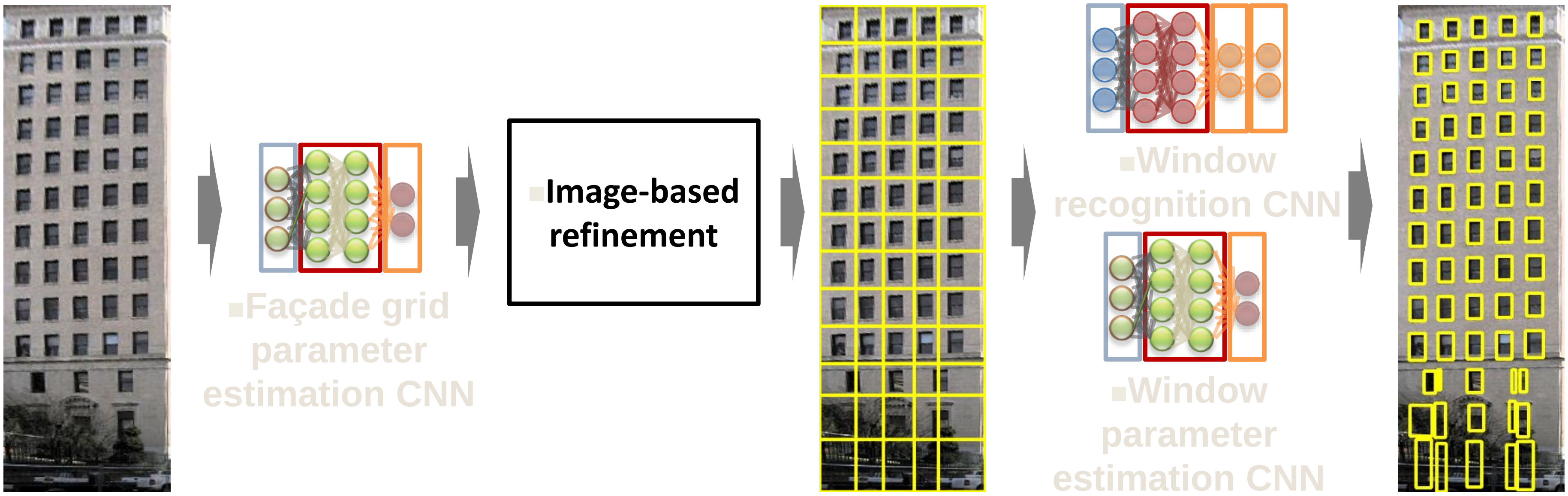
Building Mass Grammar



Façade Stage



Façade Simplification



▪Facade

▪Façade grid parameter estimation CNN

▪Image-based refinement

▪Subdivisio
n

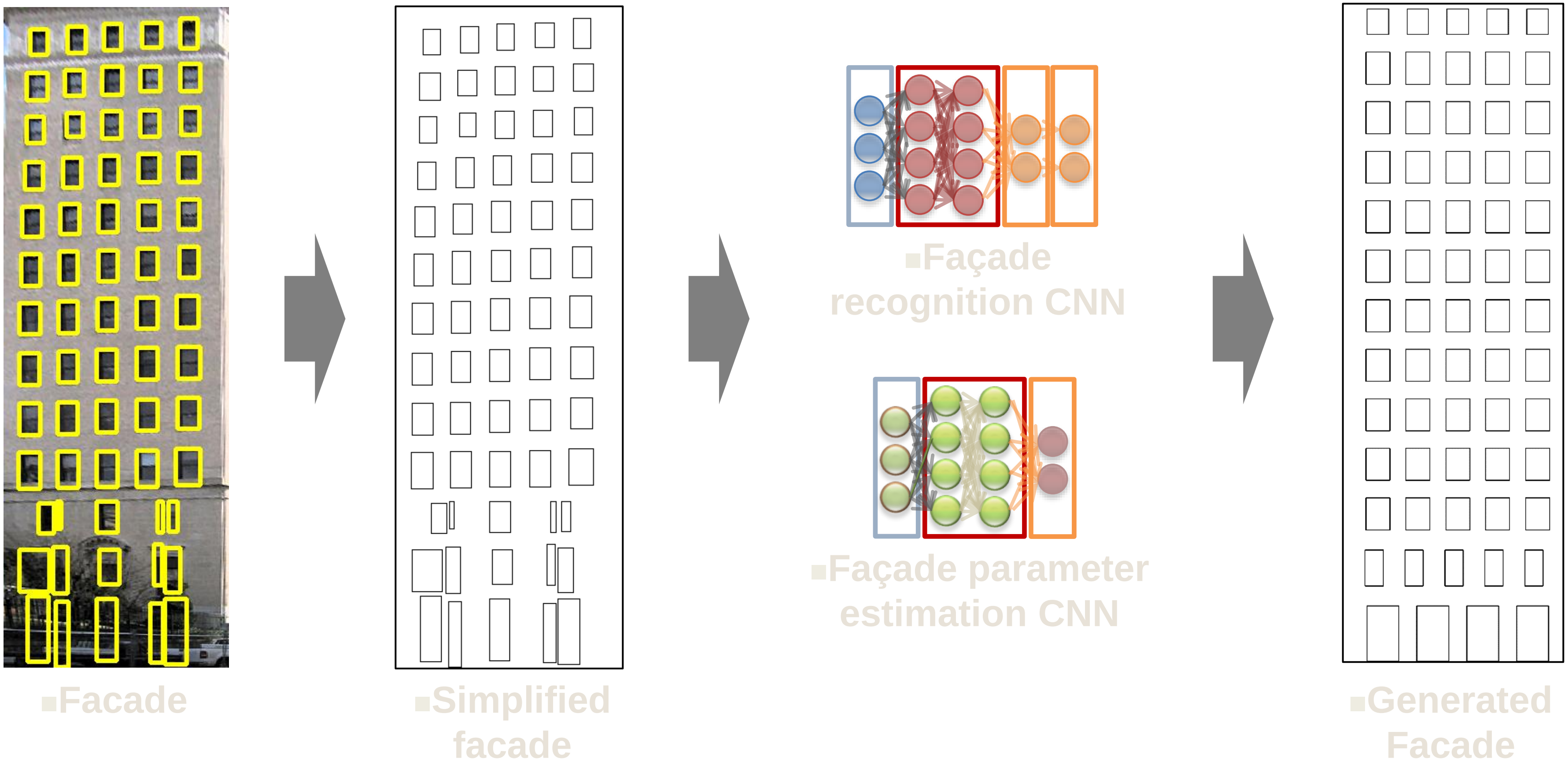
▪Window recognition CNN

▪Window parameter estimation CNN

▪(position/size)

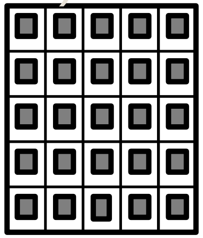
▪Detected windows

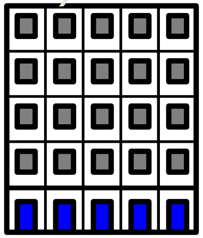
Façade Grammar Generation

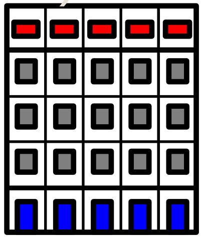


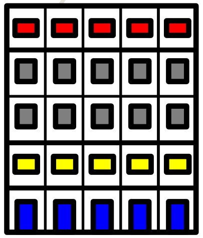
- Facade
- Simplified facade
- Façade recognition CNN
- Façade parameter estimation CNN
- Generated Facade

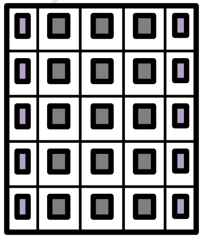
Façade Grammar

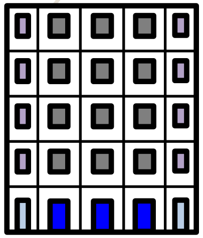
1)


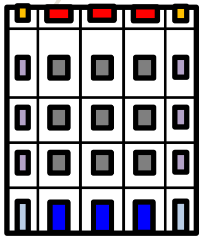
2)


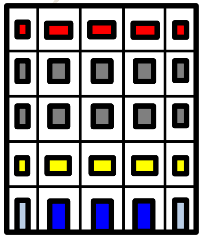
3)


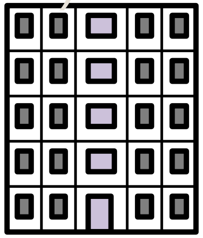
4)


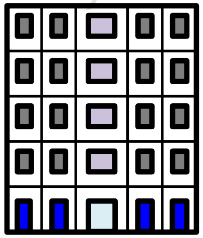
5)


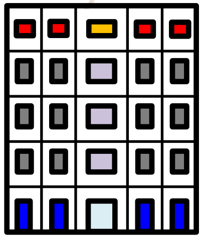
6)


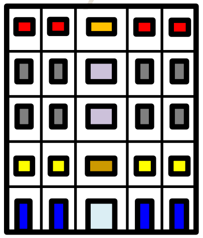
7)


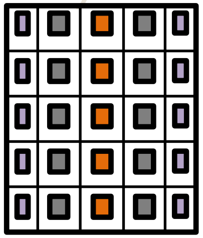
8)


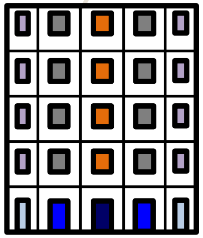
9)


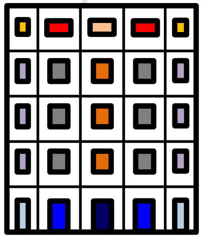
10)


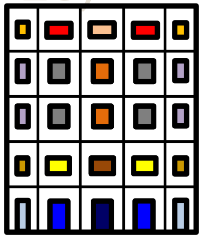
11)


12)


13)


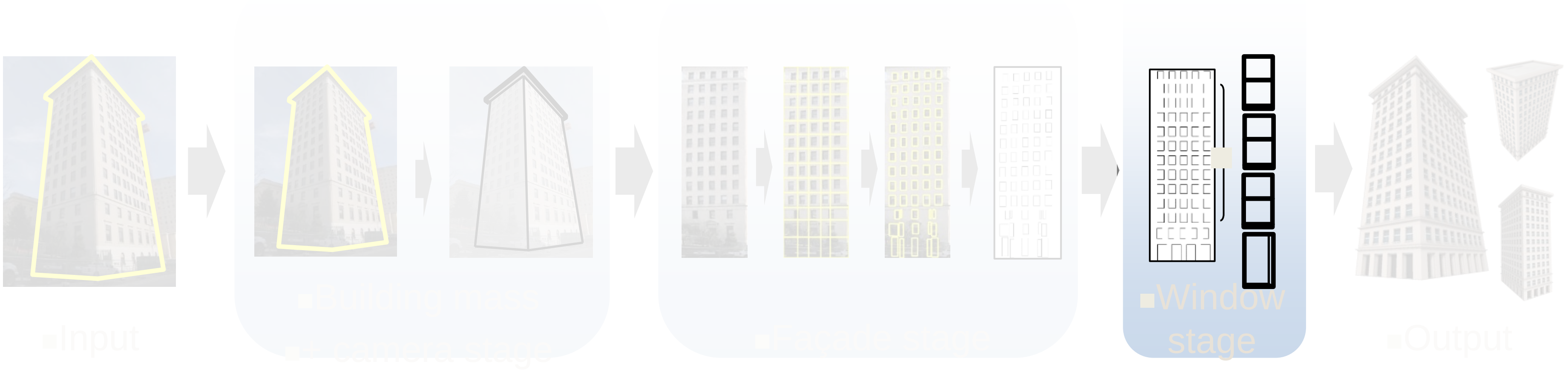
14)


15)


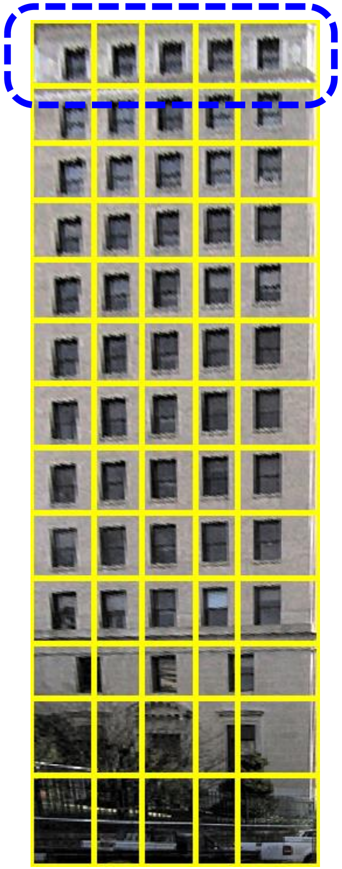
16)


- The same color denotes the same non-terminal.

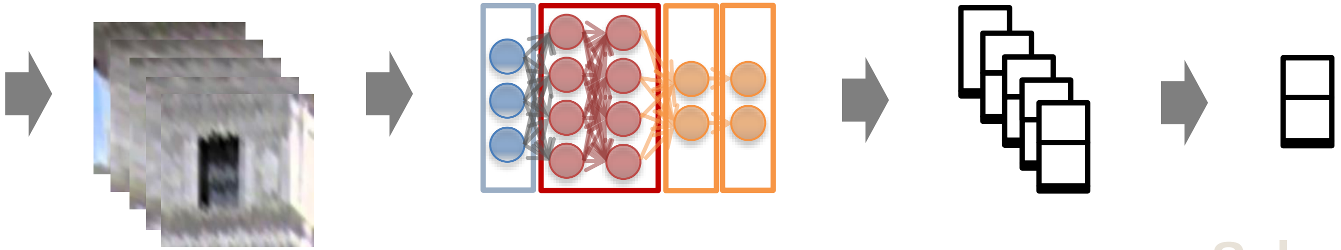
Window Stage



Window Style Recognition



- Subdivision



- Tile images
- Window style recognition CNN
- Window grammars
- Selected window grammar

- Maximum votes for the same non-terminal to select a window grammar.

Window Grammar

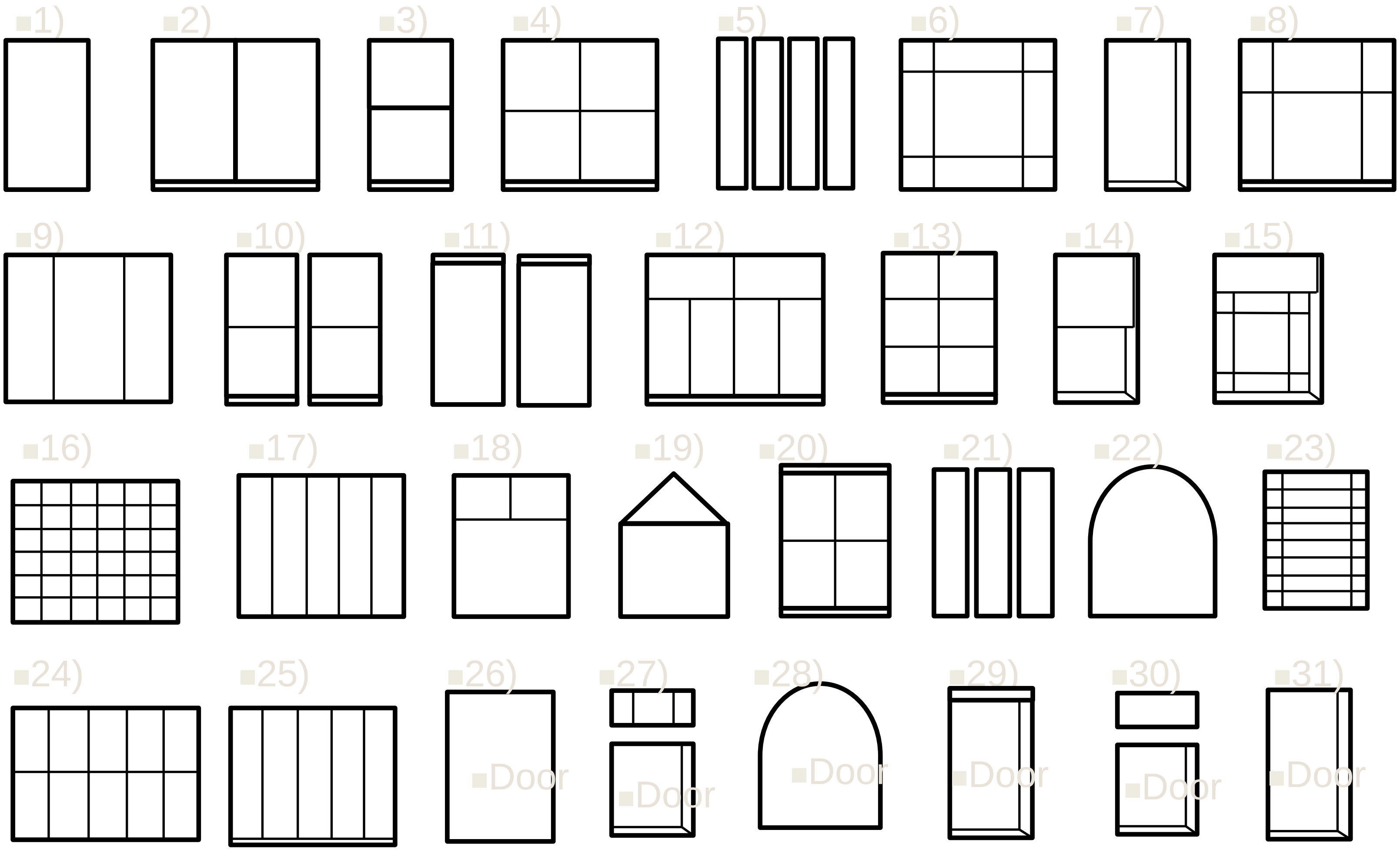


Photo to 3D Results

Aerial Images

- Still works but camera parameter estimation is less accurate due to the weaker perspective of the images.