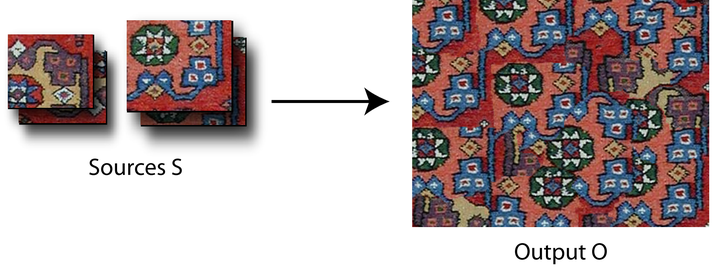
Constrained Graphcut Synthesis

Ganesh Ramanarayanan and Kavita Bala

Cornell University

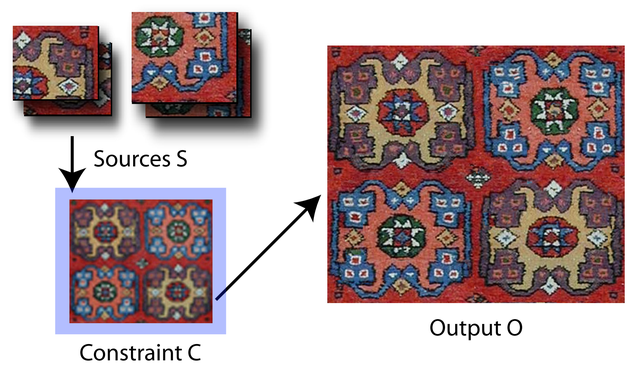
OVERVIEW

Problem



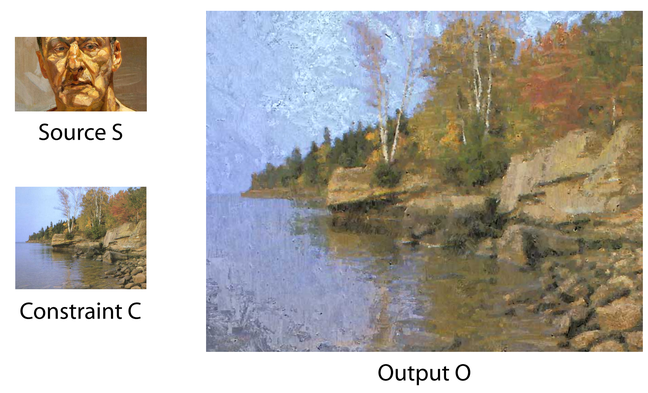
- Texture synthesis, while powerful, is **unpredictable** and **ill-defined**
- **Goal:** Robust, **controllable** texture synthesis
- **Challenges:**
  - **How** does a user control synthesis?
  - How do we define and measure **quality**?
- **Applications:** image analogies, detail synthesis, texture creation for games

Solution - CGS



- **Related work:**
  - Constrained synthesis: [Hertzmann01] [Efros01] [Ashikhmin01] [Schödl02]
  - Graphcut Textures: fast, high-quality unconstrained synthesis [Kwatra03]
- **Approach: Add constraints to graphcuts**
- **Insight: Leverage unused term of graphcut minimization framework**
- Simultaneously optimize **constraint match** and **texture seamlessness**

Contributions

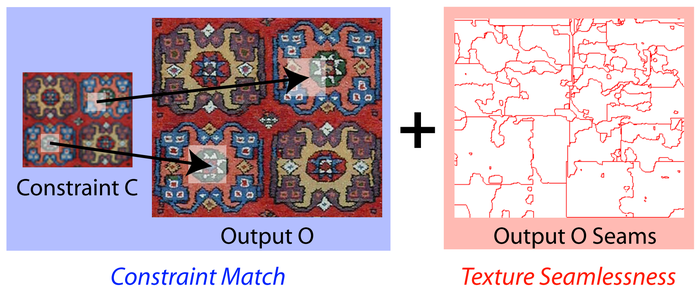


CGS run on artistic filtering example from [Hertzmann01]

- **Robustness:** Formulates synthesis as **global energy minimization**
- **Quality:** Comparable to [Kwatra03], while supporting constraints also
- **Efficiency:** Significant performance increase over previous work
- Addresses **large search spaces** in graphcut minimization

ALGORITHM

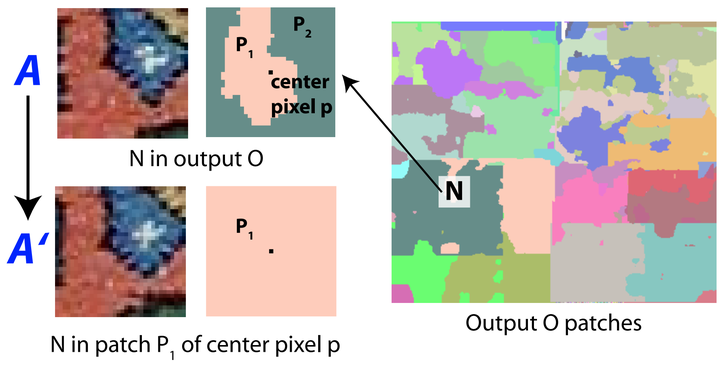
Measure of Quality



- **Constraint match:** compare neighborhoods N in constraint C and output O
- **Texture seamlessness:** difference between adjacent pixel pairs [Kwatra03]
- Sum over all neighborhoods and pairs:

$$\underbrace{\sum_{N \in O} A(C(N), O(N))}_{\text{Agreement Cost } \boldsymbol{A}} + \underbrace{\sum_{(p,q) \in O} M(p,q,O)}_{\text{Seam match cost } \boldsymbol{M}}$$

Setting up Graph Mincut

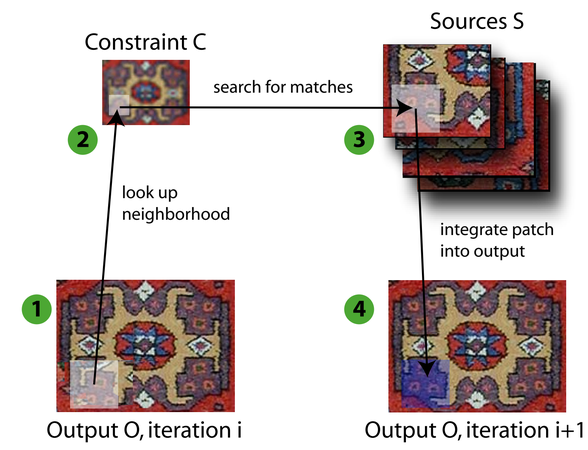


- Graphcuts cannot be applied to ***A*** since neighborhoods contain multiple patches
- Instead, use ***A'***: assume neighborhood only contains center pixel's patch
- Works well due to simultaneous optimization of seam and neighboring ***A'*** costs

$$\sum_{p \in O} A'(C(N(p)), P_p(N(p))) + \sum_{(p,q) \in O} M(p,q,O)$$

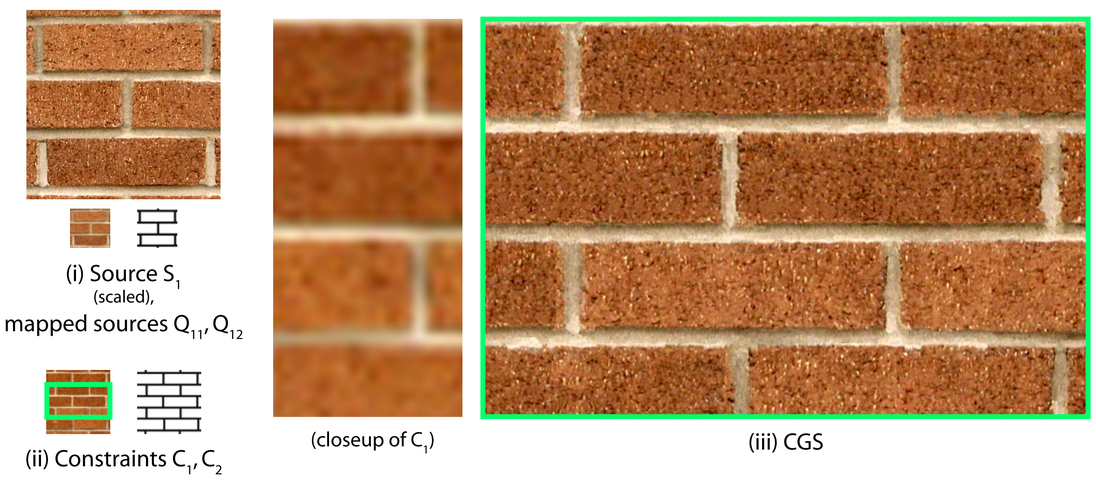
Final Objective Function E

Final Algorithm

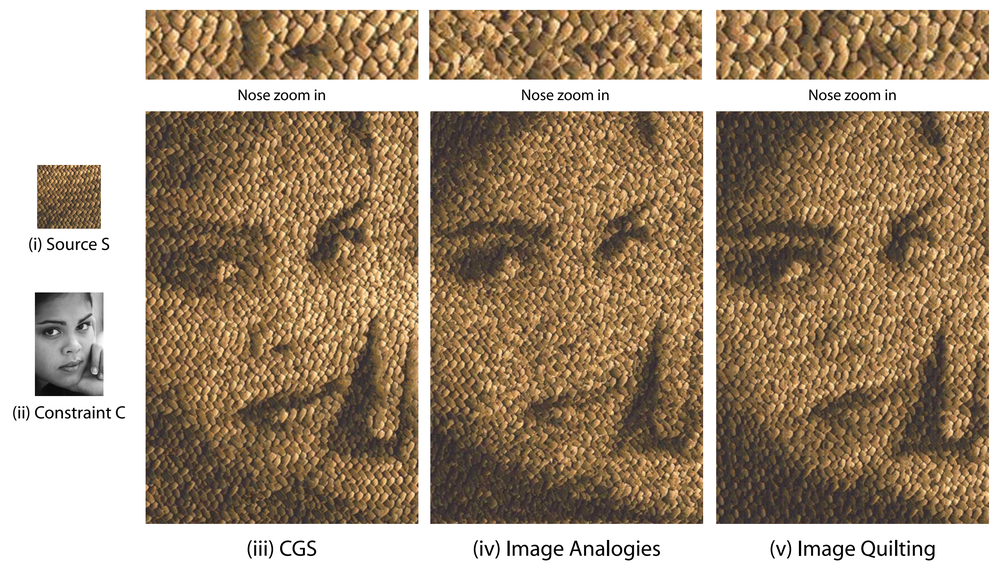


- **Step 1:** Find worst neighborhood of O
- **Step 2:** Look up corresponding neighborhood in C
- **Step 3:** Identify set of potential matching patches in S
- **Step 4:** Integrate best match into O using graph mincut with **E**
- **Termination condition:** Loop until no improvement

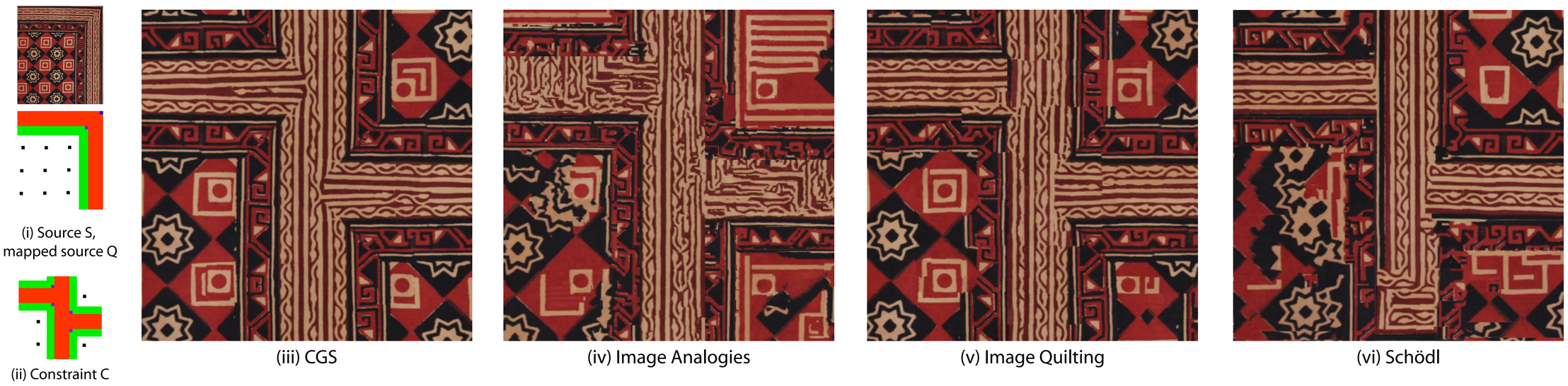
RESULTS



Novel Application: Detail Synthesis w/ Multiple Constraints



Texture Transfer



Texture-by-Numbers