Constrained Graphcut Synthesis
Ganesh Ramanarayanan and Kavita Bala
Cornell University

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

**Contributions**
- 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

**Measure of Quality**
- Constraint match: compare neighborhoods \( N \) in constraint \( C \) and output \( O \)
- Texture seamlessness: difference between adjacent pixel pairs

**Setting up Graph Mincut**
- Sum over all neighborhoods and pairs:
  \[
  \sum_{p \in O} A(C(N(p), O(p))) + \sum_{(p, q) \in O} M(p, q)
  \]

**Final Objective Function \( E \)**
- Cost function:
  \[
  E = \sum_{p \in O} A(C(N(p), O(p))) + \sum_{(p, q) \in O} M(p, q)
  \]

**Novel Application:**
- Detail Synthesis w/ Multiple Constraints
- Texture Transfer

**Texture Transfer**
- How do we define and measure quality?
- How does a user control synthesis?
- Related work: [Hertzmann01], [Efros01], [Ashikhmin01], [Schödl02]
- Graphcut Textures: fast, high-quality unconstrained synthesis ([Kwatra03])
- Approach: Add constraints to graphcut
- Insight: Leverage unused term of graphcut minimization framework
- Simultaneously optimize constraint match and texture seamlessness

**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

**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