Camouflaging an Object from Many Viewpoints

Andrew Owens[1] Connelly Barnes[2] Alex Flint[3] Hanumant Singh[4] William Freeman[1]

[1]MIT CSAIL [2]University of Virginia / Adobe [3]Flyby Media [4]Woods Hole Oceanographic Inst.

Abstract

We address the problem of camouflaging a 3D object from the many viewpoints that one might see it from. Given photographs of an object's surroundings, we produce a surface texture that will make the object difficult for a human to detect. To do this, we introduce several background matching algorithms that attempt to make the object look like whatever is behind it. Of course, it is impossible to exactly match the background from every possible viewpoint. Thus our models are forced to make trade-offs between different perceptual factors, such as the conspicuousness of the occlusion boundaries and the amount of texture distortion. We use experiments with human subjects to evaluate the effectiveness of these models for the task of camouflaging a cube, finding that they significantly outperform naïve strategies.

1. Introduction

A leopard's spots, an octopus's mottled skin – the elaborate ways that animals hide themselves have long fascinated students of biology and perception. What makes these animals hard to see?

Over the years, biologists and vision scientists have discovered many camouflage strategies. Perhaps the most familiar of these is *background matching*, which describes animals that avoid detection by having colors and textures on their bodies that are similar to those of their natural habitats. But there are many strategies that work by thwarting other visual processes, such as perceptual grouping and recognition. There are animals, for example, that obfuscate their body outlines with high-contrast markings, a strategy known as *disruptive coloration*, and others whose bodies have evolved to look like uninteresting objects such as twigs and leaves, a strategy known as *masquerade* [23].

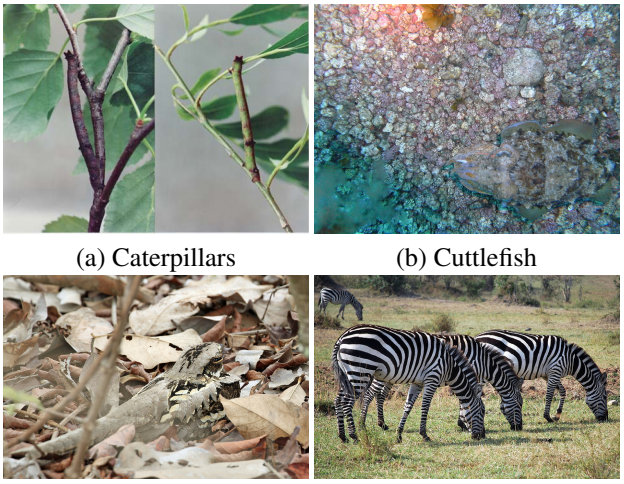
While this research has uncovered many of the principles behind camouflage, there has not been much attention devoted to the problem of automatic camouflage, where these perceptual principles are put directly into practice via algorithms. In this paper, we formulate a camouflage problem and compare several algorithms for solving it. Specifically, our goal is to determine which surface pattern will best camouflage a 3D object from the many different viewpoints that one could see it from (Figure 1).

Figure 1: Four views of a camouflaged box. We studied algorithms for designing the pattern that is drawn on the box's surface: here we show the model *Interior MRF* (Section 4.3), which tries to hide seams at the box's occlusion boundaries and on its interior. For demonstration purposes, we printed the pattern on paper and wrapped it around a real box; in the rest of the paper we render a synthetic box.

While camouflage is interesting as a scientific topic, there are also many applications. One of these is to hide unsightly objects such as utility boxes, an idea that has been playfully explored by public artists [4]. Camouflage can also be used to observe a scene while remaining hidden, and bird-watchers and hunters often conceal themselves with camouflaged clothing, or by staying in special enclosures.

Camouflage is intimately connected with human and computer vision. While computer vision provides detection strategies, *e.g.* for objects [11] and boundaries and salient regions [15], camouflage provides strategies to avoid being detected by these cues. Thus we view camouflage as the inverse problem of object detection. That is, we wish to solve the *object non-detection* problem: to make an object whose appearance is not detectable.

1063-6919/14 $31.00 © 2014 IEEE
DOI 10.1109/CVPR.2014.350



(a) Caterpillars (b) Cuttlefish

(c) European Nightjar (d) Zebras

Figure 2: Animals use many strategies to trick the visual system. (a) Caterpillars masquerade as twigs [18]; (b) Cuttlefish change their skin pattern and use muscles to add geometric bumps or ripples; (c) A European Nightjar uses background matching and disruptive coloration; and (d) there are many hypotheses about the function of Zebras' stripes [21].

Our technical approach is inspired by the remarkable camouflage abilities of cephalopods such as octopuses and cuttlefish (Figure 2). These animals observe the surrounding environment and then display complex patterns on their skin for concealment [5]. By analogy, we modularize the camouflage problem into two stages: capturing the scene and computing a camouflage pattern. In the first stage (Section 3), we take photographs from the views that we wish to hide the object from, and then we place a synthetic 3D object into the scene (always a box shape in our experiments). In the second stage (Section 4), we decide how to color the surface texture. This requires modeling and making trade-offs among the perceptual cues that could give away the presence of the object. Our algorithms include a naïve model as well as more sophisticated models that balance between these factors. Finally, we use psychophysical experiments on Amazon Mechanical Turk (Sections 5, 6) to verify that our camouflage algorithms hide objects significantly better than a naïve strategy.

Our technical contributions include: (1) new proposed models for camouflage of 3D objects; (2) a dataset of 37 scenes that can be used to objectively compare camouflage models; and (3) a study methodology for comparing camouflage methods according to human visual perception.

2. Related work

The related work spans biological camouflage, computer vision, and computer graphics.

Biological camouflage The algorithmic models we discuss are based on the strategy of background matching, but there are many other camouflage strategies found in nature. These strategies include disruptive coloration, which attempts to hide the object's true outline, and *countershading*, which attempts to make the visual cue of illumination less detectable by shading the bottom of an animal more lightly than the top. Early work in this area, such as the seminal work of Hugh Cott [7], provided extensive descriptions of these phenomena, while recent work has tested these ideas with experiments. These strategies, and the state of current research, are discussed extensively in a recent textbook on animal camouflage [23].

Computational camouflage There has been other work that applies computational methods to problems in camouflage. For example, Reynolds [20] uses genetic programming and a human in the loop to find effective camouflage patterns for textured 2D backgrounds. There is also ongoing work that estimates effective camouflage patterns using crowdsourced camouflage games [1]. In contrast, our work attempts to hide a 3D object from many views simultaneously, and we focus on techniques that are fully automatic. And rather than finding camouflage patterns that work well in multiple places, ours are tailored for specific 3D locations. Additionally, there has been work that uses simple computer vision models to "break" camouflage [24], and work in computer graphics on "camouflage images" [6] allows one to insert hidden images into a picture.

Texture and MRFs Our work takes inspiration from texture synthesis [14, 10, 17]. In particular, we introduce camouflage models based on Markov random fields (MRFs) that are reminiscent of those used in shift-map image editing [19], a method that can be used for texture synthesis or other image manipulations. Our MRF energy function is also similar to that of seamless montage [13], which aligns pictures on a 3D surface while minimizing seams.

The problem of image inpainting [2, 8] is closely related to, yet distinct from, our camouflage problem. With inpainting, one removes an object from an image by replacing it with a plausible background texture. In our case, due to our capturing procedure we already know the background, so we could exactly solve the single-view inpainting problem. Furthermore, we wish to camouflage a 3D object from multiple viewpoints, which raises a problem that is not usually addressed in inpainting methods: that of resolving the many conflicting opinions of very different images.

3. Capturing scenes for camouflage

To camouflage an object, we first need to capture images of the surrounding environment. For each scene, we capture approximately 10-25 photographs, each representing a different viewpoint that the object should be hidden

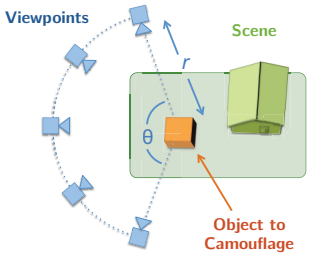


Figure 3: Our capture setup. A real scene contains an object that is to be camouflaged. We capture the scene without the object from a wide angular range θ of camera viewpoints. In some scenes we also vary the height from the ground and object-camera distance r.

from. During capture, no camouflaged object is actually placed in the scene, and only pictures of the background are collected. An example capture setup is shown in Figure 3. Generally, these pictures are taken from similar distances to the target object, but with varying spherical angles. Finally, we estimate the camera pose using structure from motion [22] and place a synthetic 3D object into the scene using a simple user interface. In our studies, this object is always cube-shaped (more precisely, a rectangular cuboid).

We focus on computing effective camouflage patterns, and we defer to other studies for many practical tasks of artificial camouflage. These include the task of displaying the pattern (*e.g.* on a screen or printed surface), and concealing the object from shadows and illumination gradients (or removing these with lighting). Furthermore, objects with other shapes may pose special challenges, *e.g.* texture distortion due to curved surfaces or concavities. Many of these issues have been studied in 3D texture synthesis work [16].

4. Our camouflage models

In this work, we study *background matching*, the camouflage strategy that hides an object by making it look like whatever is behind it. Since it is impossible to make an object look exactly like the background from every viewpoint, there are trade-offs that must be made. How important is it to conceal the object's occlusion boundaries? Are some viewpoints more important to be concealed from than others? We address these questions, designing models that explore these different aspects of background matching.

4.1. Naïve model: mean coloration

One simple solution is to color the object so that it matches the background's coarse color statistics. Thus for the *Mean* model, we project each image onto the cube and then estimate the mean color at each texel (*i.e.* at each position on the object's surface). This algorithm captures the coarse colors of the background, but the result is blurry and often conspicuous (Figure 4a).

4.2. Random and greedy viewpoint models

Rather than matching coarse background statistics, we consider the alternative of matching a small number of viewpoints more precisely. To motivate this idea, suppose that we colored the object by projecting an image onto it. This would make the object completely invisible from the chosen viewpoint; however, it would often be hidden from *nearby* views as well.

This leads naturally to an algorithm. We choose an image and project it onto the cube, filling in the faces that are visible from that viewpoint. We then choose another view and project it onto the uncolored cube faces, repeating until the whole cube is filled in.

With the goal of understanding how the choice of viewpoint matters, we consider two versions of this model. The first of these models, *Random*, simply projects images onto the cube in a random order. However, this model leads to textures that appear stretched or distorted (Figure 4b). For example, if we were to project an image onto a face that was very tilted with respect to the viewing direction, and then viewed the result from a nearby position where the face was less tilted, then we'd see significant distortion artifacts; the face may even be occluded from many nearby views. On the other hand, if we were to project an image onto a fronto-parallel face, then the texture would look very similar from nearby views. We call this *viewpoint stability*. Thus for our second model, which we call *Greedy*, we only project an image onto faces that are observed from an angle of less than 70°. And rather than projecting the images in a random order, we always pick the viewpoint that sees the most faces from such an angle[1].

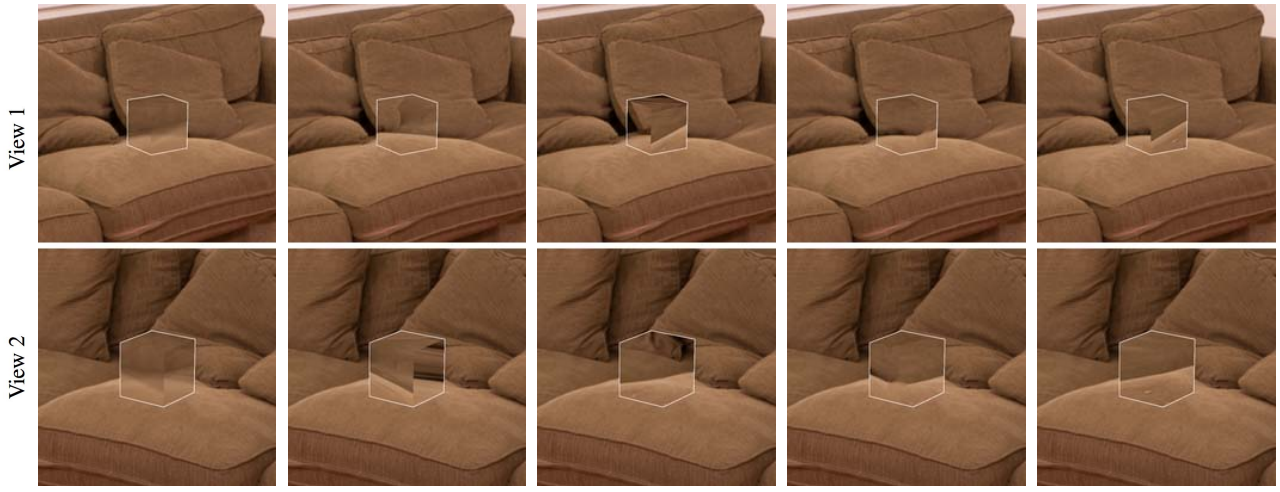
We note the conceptual similarity between viewpoint stability and the generic viewpoint principle [12]. Informally, we prefer to project images that see "generic" views of the surface texture, rather than "accidental" ones.

4.3. MRF models for hiding boundaries

Often what gives away the presence of an object is a mismatch in texture along occlusion boundaries. These occlusion boundaries pose a special challenge for camouflage, because from different viewpoints the occluded background may change drastically. Also, since the object is often composed of multiple textures, the seams between textures on the interior of the object are another potentially important detection cue (both cues can be seen in Figure 4c).

We develop a Markov random field (MRF) framework that trades off between avoiding these three detection cues: occlusion matching, interior seams, and viewpoint stability. Our formulation is similar to MRF-based texture synthesis and hole-filling methods [19, 13]. We divide the mesh sur-

[1] If a face is never seen from a shallow angle, then we allow it to be colored by any view. Ties between viewpoint choices are broken randomly.



(a) Mean (§ 4.1) (b) Random (§ 4.2) (c) Greedy (§ 4.2) (d) Interior MRF (§ 4.3) (e) Boundary MRF (§ 4.3)

Figure 4: Model comparison. Two views of cubes synthesized by each model (top and bottom). The Mean coloration model (a) produces a blurry result. The Random projection model (b) can produce textures that look distorted; the Greedy projection model (c) picks undistorted textures that may not match the image content well. These two methods are randomized, and for illustration purposes we picked random seeds where these failure modes occurred. The Interior MRF (d) trades off between concealing occlusion boundaries, texture distortion, and interior seams. The Boundary MRF (e) chooses a pattern that matches the background for a cluster of viewpoints, allowing seams to occur only at face boundaries.

face into texels (256×256 texels per face), and the inference problem is to label each texel with an index (v, p) where v and p identify a view and a pixel on that view (which defines the texel's color). In our first model, we connect the texels as a grid graph and add edges between texels on neighboring faces (Figure 5a), while in our second model we assign a label to each face and connect adjacent faces.

We use a label space similar to that of [13] (Figure 5b): if a texel is observed in view v at pixel p, then we add $\{(v, p + \Delta_p) \mid \Delta_p \in T\}$ to the label space, where T is a set of *translations*. In our models, we use either $T = \{(0, 0)\}$, in which case labels simply correspond to views, or we use a 3×3 grid. We minimize the energy function

$$D(\{x_i\}) = \sum_i E_i(x_i) + \sum_{i,j} E_{ij}(x_i, x_j), \qquad (1)$$

where x_i is the label for texel i, and E_i and E_{ij} are the data and smoothness costs.

For the data cost, we trade off between matching occlusion boundaries and choosing a texture that comes from a stable view of the face:

$$E_i(x_i) = E_i^O(x_i) + E_i^S(x_i), \qquad (2)$$

where E_i^O and E_i^S are the occlusion-matching and stability costs. We define the occlusion penalty to be

$$E_i^O(x_i) = \frac{1}{m} \sum_{j=1}^{m} w_{ij} ||c(x_i) - c_{ij}||, \qquad (3)$$

where $c(x_i)$ is the label's corresponding color, c_{ij} is the color observed at the projection of texel i into image j, and m is the total number of views. If the texel is near an occlusion boundary for a particular view, then the normalization factor w_{ij} is set to n_t/n_j, where n_t is the total number of texels and n_j is the number of occlusion texels in view j; when the texel is on the interior we set $w_{ij} = 0$. We consider a texel an occlusion texel if it is within a certain distance of an occlusion edge (we use 10% of the cube side length as the threshold), and we blur the input images for robustness to high-frequency textures.

For the texture stability cost, we estimate how distorted the texture will appear when seen from other views:

$$E_i^S(x_i) = \frac{1}{m} \sum_{i'=1}^{m} \min(\rho(J(v(x_i), i', f_i)), \tau). \qquad (4)$$

Here $v(x_i)$ is the view that the texel has been assigned through its label, and f_i is the texel's face. The term $J(i, i', f)$ is the Jacobian at the center of face f of the change in pixel coordinates (p'_x, p'_y) at view i' relative to pixel coordinates (p_x, p_y) in view i: $J(i, i', f) = \partial(p'_x, p'_y)/\partial(p_x, p_y)$. The stability function ρ sums over both of the eigenvalues of J the penalty $\rho^*(\lambda) = \alpha \max(\lambda - \gamma, 0)^2$, where λ is an eigenvalue. We set the parameters to discourage stretching factors that arise due to very tilted views, and to limit the contribution due to any single pair of views, setting $\gamma = 2$, $\alpha = 20$, and $\tau = 60$.

To investigate the trade-offs between these different de-

tection cues, we study two variations of this MRF model. We call the two resulting models the *Interior MRF* and the *Boundary MRF*. The two models share the same data costs but have different strategies for achieving smoothness on the cube interior. Results are shown in Figure 4.

Interior MRF Our first MRF model attempts to hide all texture edges on the interior of the cube. We define a smoothness cost between adjacent texels x_i, x_j that penalizes seams, similar to the smoothness term in [19]:

$$E^I_{ij}(x_i, x_j) = \alpha_I \sqrt{||c(x_i) - \tilde{c_i}||^2 + ||c(x_j) - \tilde{c_j}||^2}, \quad (5)$$

where $\tilde{c_i}$ is the color texel i would have if it were given the label x_j, and similarly for $\tilde{c_j}$. In more detail, suppose that in view v the texels i and j project to pixels p_i and p_j respectively, and that texel j is labeled with $(v, p_j + \Delta_p)$. Then the prediction $\tilde{c_i}$ is the color at pixel $p_i + \Delta_p$ in view v. In particular, if the two labels come from the same image and use the same translation, then the cost is zero. To allow the MRF to synthesize texture when hiding seams, we enable label translations, setting $T = \{-d, 0, d\} \times \{-d, 0, d\}$ where $d = 30$ pixels. To balance the seam cue with the occlusion cues, we set $\alpha_I = 32$. We use graph cuts with α-expansion [3] to find a labeling with this model.

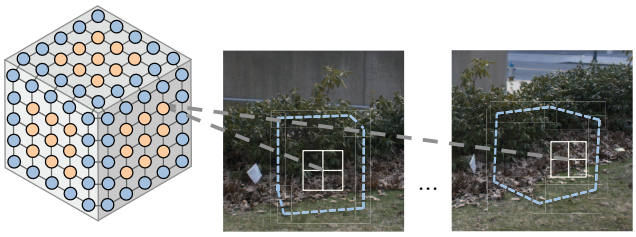
Boundary MRF It is natural to ask whether some interior seams are more conspicuous than others. We explore this idea in our second MRF model, the *Boundary MRF*, which minimizes the same data costs as Interior MRF but only allows seams to occur at boundaries between faces (whereas the Interior MRF allows them to occur anywhere on the interior). The assumption behind this model is that for polyhedral shapes, face boundaries are relatively "safe" places to put seams: a seam on the interior of a face will be visible from every view that sees the face, but a boundary seam will only be visible to views that see both faces.

To remove seams on faces, we require all of the texels on a face have the same label. Thus for efficiency reasons we assign a label to the whole face, rather than to individual texels[2]. The data cost for one of these face "supernodes" is the cost of assigning the label to all of its texels:

$$E_f(x_f) = \sum_{i \in F_f} E_i(x_f), \quad (6)$$

where x_f is the label for this face, and F_f is the set of texels belonging to this face. Adjacent faces are connected together, forming a six-node MRF. For the smoothness cost, we use a very large Potts cost, penalizing adjacent faces with different labels, which effectively requires the solution to have minimal seams. Since we do not allow seams on faces, we disable pixel translations, setting $T = \{(0, 0)\}$. We solve for the optimal solution using brute-force search.

[2]A very similar (but harder to optimize) model can be obtained by modifying the smoothness cost in Interior MRF: adding a large Potts cost for texels on the same face and a smaller Potts cost for between-face edges.



(a) MRF (b) Label space for one texel

Figure 5: The Interior MRF is structured as a grid graph where each texel is a variable (a). Texels near face boundaries (in blue) have special occlusion-boundary costs. Each texel is labeled (b) with a pixel from one of the views, plus a shift that comes from a grid; this grid is centered on the texel's projection in each view. In Boundary MRF, we use the same data costs, but texels on each face are combined into a single node, and we do not allow shifts.

For typical scene configurations (and cube-shaped objects), the solutions contain projections from just two representative viewpoints: one view is usually projected onto three faces, and another is used for the rest. In this way, the model is similar to the Greedy model. Both models choose a small number of representative views to project onto the cube, but the Boundary MRF decides on these views using explicit data costs instead of viewing-angle heuristics.

5. Psychophysical study design

We evaluate our camouflage algorithms with psychophysical studies on Amazon Mechanical Turk. Our experimental setup resembles work in visual search [25] and also studies that compare the effectiveness of different camouflage patterns [1, 9]. We show subjects a sequence of pictures: some contain a box and some do not. They are instructed to press the *y* key if they see a box and *n* if they do not. If they answer *y*, they must click on the box. We use a distance threshold to test whether their click is near the box (8% of the image width). They are also told that after 45 seconds, it will no longer be possible to respond. After the user responds or times out, we outline the box or state that there was no box. To make users comfortable with the setup, we include a short practice (6 images).

To avoid possible interference effects, we show each subject examples from only one algorithm. Since the task is easy if one knows the location of the box, we only show users one viewpoint per scene, which is chosen at random for each user. We randomly (50% chance) pick whether the image contains a box. To ensure that the image being shown was not used as an algorithmic input, we use a "leave-one-out" procedure. For each image, we synthesize a camouflaged box using all other photos of the scene and then render from the held-out view. To exclude poorly performing

Mechanical Turk workers, we also include eight easy images, and exclude users who perform poorly on them.

We use two evaluation metrics. First, we measure *confusion rate*, the probability of a subject answering incorrectly. Second, we measure *time-to-find*, the amount of time it takes the user to press y, given that there actually is a box in the scene and that the user subsequently clicks on it. A good camouflage algorithm would thus cause a user to have a high confusion rate and large time-to-find. In our evaluation, we "pool" these values per subject: we compute the subject's median time-to-find and their confusion rate, and we test whether users put into one algorithmic treatment perform differently than those put into a different treatment.

6. Results

6.1. Quantitative results

We evaluated the five models described in Section 4 using our study design. We found these methods all outperform the naïve model on both the time-to-find and confusion rate evaluation metrics (Table 1). Furthermore, we found that the MRF methods significantly outperform the random projection methods on time-to-find, while all of the best-performing methods have similar confusion rates.

For confusion rate, we tested for significance using a two-sided t-test, and for the (long-tailed) time-to-find metric we used the two-sided Mann-Whitney U. Since we performed multiple tests (compared every algorithm with every other on two metrics) we used the Holm-Bonferroni method to control the number of false discoveries at the significance level of 0.05 (we essentially use a significance threshold of 0.01 after correction).

Both MRF-based methods had significantly higher time-to-find than the projection-based methods. In particular, the median time for Boundary MRF was more than a half-second greater than that of Greedy ($p < 10^{-6}$ for the time-to-find comparison between these models), and Interior MRF outperformed Greedy by more than 200ms ($p = 0.006$). However, the difference between the two MRF models on time was not significant ($p = 0.027$).

While these three models differed on time, there was not a large difference between their confusion rates. The MRF-based methods' confusion rates were about 2% higher on this metric than Greedy, but this difference was not significant ($p = 0.011$ when comparing Interior MRF and Greedy). Meanwhile, Interior MRF and Boundary MRF methods had almost identical confusion rates, though the former's was insignificantly higher. Between the two projection-based methods, we found that Greedy significantly outperformed Random on confusion rate ($p = 0.003$) but not on time ($p = 0.394$). All methods were significantly better than Mean on both metrics ($p < 10^{-6}$ for all comparisons). See the supplemental material for full statistics.

Model	Confusion	Time to find	Overall time	n
Uniform	0.03 ± 0.00	$1.38\text{s} \pm 0.03$	$1.42\text{s} \pm 0.03$	327
Mean	0.09 ± 0.00	$1.70\text{s} \pm 0.04$	$1.74\text{s} \pm 0.03$	328
Random	0.25 ± 0.01	$2.59\text{s} \pm 0.08$	$2.59\text{s} \pm 0.06$	288
Greedy	0.28 ± 0.01	$2.66\text{s} \pm 0.11$	$2.72\text{s} \pm 0.08$	309
Interior MRF	$\mathbf{0.30 \pm 0.01}$	$2.90\text{s} \pm 0.10$	$2.92\text{s} \pm 0.08$	299
Boundary MRF	0.30 ± 0.01	$\mathbf{3.25\text{s} \pm 0.13}$	$\mathbf{3.27\text{s} \pm 0.09}$	284

Table 1: Quantitative results. Higher is better. We report the mean confusion rate and median time-to-find (Section 5). We also include the *overall time*, which is the same as time-to-find but with the median taken across all responses rather than per user, and n, the number of users (each sees 37 images). The baseline Uniform gives the object a solid color, acting as a lower bound on performance. Confusion rates include both false positives and negatives (for comparison, the chance of missing a true cube for Interior MRF is 46%; see supplemental material for details). Standard errors for time were computed using bootstrapping.

The fact that Boundary MRF and Interior MRF improve over Greedy on time suggests that occlusion boundary mismatches and texture distortion are useful cues to conceal for background matching. As one possible explanation for the similar confusion rates, we note that all methods are based on the same principle: that projecting a view onto the cube will conceal it from nearby views. In some cases, there may be few views that conceal the cube well. In this situation, the observer may miss the cube only when their viewpoint is close to the projected one; otherwise the imperfections will be obvious enough that they will find the cube eventually.

6.2. Qualitative examples

We show representative results in Figure 6, and we show multiple viewpoints for a small number of scenes in Figure 7. To illustrate the differences between the models, we also compare the results of different models in Figure 8.

7. Discussion

Our work suggests that the best background matching models must trade off between concealing different cues, such as texture distortion (Greedy), occlusion matching (Boundary MRF), and hiding interior seams (Interior MRF). While in general these goals appear to be in conflict, particular scenes show that one model will often excel over its competitors. For example, in scenes where the object is close to a backing wall or floor (Figure 8a), occlusion cues are relatively easily to hide. In other scenes, thwarting these occlusion cues is not enough, and interior seams or texture distortion may be more important (Figure 8b).

We see our paper as opening two directions of future work. First, while we have focused on creating patterns that are effective in psychophysical studies, the task of convincingly displaying these patterns in the real world has challenges of its own, such as lighting and printing limitations.

Figure 6: One viewpoint from 20 (of 37) scenes used in our study. Camouflaged cubes have been placed in each one using the camouflage model Boundary MRF (Section 4.3). Can you find them? Zooming is required. We sampled scenes randomly after filtering by aspect ratio, and sorted them by their confusion rates from the psychophysical study (scenes with low confusion rate shown first). We chose views where the cube is easier to find (many omitted views are substantially harder). We used all of the viewpoints to compute the camouflage patterns, without the leave-one-out procedure used in the study.

Figure 7: Multiple views for selected scenes, camouflaged using the Boundary MRF model.

As an initial experiment in this area, we printed a camouflage pattern and wrapped it around a real box (Figure 1). The result has visible color mismatches and shading effects. Another direction is to explore different biological principles of camouflage. In this work we focused on background matching, but there are many other strategies for camouflage in biology (Figure 2), such as disruptive coloration, masquerade, actively emitting light, and changing shape.

In conclusion, we have presented and tested new models that camouflage an object from multiple viewpoints. We performed human studies and found that our models improve significantly over a naïve camouflage model. Our results also provide insight into the cues that are important for camouflage. They suggest that projecting viewpoints onto

(a) Boundary MRF *vs*. Greedy (b) Interior MRF *vs*. Boundary MRF (c) Greedy *vs*. Boundary MRF

Figure 8: Result comparison. We show cases where one model (left image) produces a better camouflage than another (right image), as measured by confusion rate. In (a), the Boundary MRF chooses textures that match a planar background and a textured scene well. In (b), the Interior MRF stitches together textures that match the backgrounds well, while the Boundary MRF introduces warping and seam artifacts. In (c), the Boundary MRF chooses a texture that contains a conspicuous table leg, which contributes little to the cost but is semantically important; in the second example it hides the box from one cluster of viewpoints (pointing parallel to the log) but makes it conspicuous to others (those facing the log).

the cube is better than coarsely matching colors, and that it can take significantly longer for someone to find the object if these views are chosen well. Finally, our results suggest that matching occlusion boundaries and modeling texture distortion results in better camouflage, and that hiding all interior seams may be less important. Our database, as well as the source code for the algorithms and psychophysical experiments, will be available to other researchers.

Acknowledgments. We thank Yair Weiss and the members of Ruth Rosenholtz's group for the helpful discussions. This work was supported by NSF CISE/IIS award 1212928 and an NDSEG Fellowship to A.O. Photo credits for Fig. 2: [18] and Flickr users *84884728@N03*, *giselaglb*, *schinker*.

References

[1] Where is that nightjar? http://nightjar.exeter.ac.uk/where-is-that-nightjar. Accessed: 2014-04-12. 2, 5

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *ACM Trans. Graphics*, 2000. 2

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. PAMI*, 2001. 5

[4] J. Callaghan. Public art projects. http://joshuacallaghan.com/publicart.htm. Accessed: 2014-04-12. 1

[5] C.-C. Chiao and R. T. Hanlon. Cuttlefish camouflage: visual perception of size, contrast and number of white squares on artificial checkerboard substrata initiates disruptive coloration. *Journal of Experimental Biology*, 2001. 2

[6] H.-K. Chu, W.-H. Hsu, N. J. Mitra, D. Cohen-Or, T.-T. Wong, and T.-Y. Lee. Camouflage images. *ACM Trans. Graphics*, 2010. 2

[7] H. B. Cott. Adaptive coloration in animals. 1940. 2

[8] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 2004. 2

[9] I. C. Cuthill and A. Székely. Coincident disruptive coloration. *Phil. Trans. of the Royal Society B: Biological Sciences*, 2009. 5

[10] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *ACM Trans. Graphics*, 2001. 2

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. PAMI*, 2010. 1

[12] W. T. Freeman. The generic viewpoint assumption in a framework for visual perception. *Nature*, 1994. 3

[13] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or. Seamless montage for texturing models. In *Com. Graph. Forum*, 2010. 2, 3, 4

[14] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *ACM Trans. Graphics*, 1995. 2

[15] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *CVPR*, 2007. 1

[16] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, and T.-T. Wong. Solid texture synthesis from 2d exemplars. *ACM Trans. Graphics*, 2007. 3

[17] S. Lefebvre and H. Hoppe. Parallel controllable texture synthesis. In *ACM Trans. Graphics*, 2005. 2

[18] M. A. Noor, R. S. Parnell, and B. S. Grant. A reversible color polyphenism in american peppered moth (biston betularia cognataria) caterpillars. *PloS one*, 2008. 2, 8

[19] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *ICCV*, 2009. 2, 3, 5

[20] C. Reynolds. Interactive evolution of camouflage. *Artificial life*, 17(2):123–136, 2011. 2

[21] G. D. Ruxton. The possible fitness benefits of striped coat coloration for zebra. *Mammal review*, 2002. 2

[22] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Trans. Graphics*, 2006. 3

[23] M. Stevens and S. Merilaita. *Animal camouflage: mechanisms and function*. Cambridge University Press, 2011. 1, 2

[24] A. Tankus and Y. Yeshurun. Convexity-based camouflage breaking. In *Pattern Recognition*, 2000. 2

[25] A. Toet. *A high-resolution image data set for testing search and detection models*. TNO Human Factors Research Institute, 1998. 5