

Supplementary Material

OPENSURFACES: A Richly Annotated Catalog of Surface Appearance

1 Overview

In this supplementary material, we provide further details for each of our Amazon Mechanical Turk (MTurk) tasks, and on our implementation.

The data from all of our experiments is fully open and available online at <http://opensurfaces.cs.cornell.edu/>. Statistics about the experiments and the results are also hosted online, and they will be continually updated as the database grows.

Task descriptions include anonymized user feedback. Such feedback is quoted with the original spelling and capitalization.

2 Downloading images

Images were downloaded from Flickr. They were selected according to several criteria. A complete description of the selection process is below.

- Queries were constructed as a *scene category + tag + "-hdr"*.
 - Scene categories: kitchen, bedroom, bathroom, living+room, dining+room, family+room, attic, pantry, garage, cellar, game+room, laundry+room, home+office, home+workshop, foyer, trophy+room, home+theater, closet, staircase, wine+cellar, hallway.
 - Tags: +interior+design, +remodel, +showroom, +architecture, +after, +renovation, +real+estate.
 - “-hdr”: we avoided photos that use HDR tonemapping since we found that such images often aimed for artistic effects rather than realistic appearance.
- The extra tag is discarded and photos are grouped by scene. Duplicates are detected by md5 hash of the file (to facilitate running scripts multiple times). The extra tags are used only to encourage nicer photos; we found that these tags resulted in images that were less cluttered and aimed more at showing off the space, rather than showing people or other activities.
- We limited ourselves to Creative Commons photos that allow “sharing” and “remixing”.
- We downloaded *every* image between 2003 Jan 01 and 2013 Jan 06, for a total of 1,099,277 images (1.4 terabytes).
- Automatic filtering of images: each image must satisfy the following conditions,
 - JPEG format
 - ≥ 6 megapixel resolution
 - ≤ 32 megabyte file size
 - at least one pixel has color (defined as minimum difference between RGB channels ≥ 10)
 - focal length specified in an EXIF header (obtained with the `jhead` utility program)
 - the camera model exists in a camera database.

3 Filtering images by scene

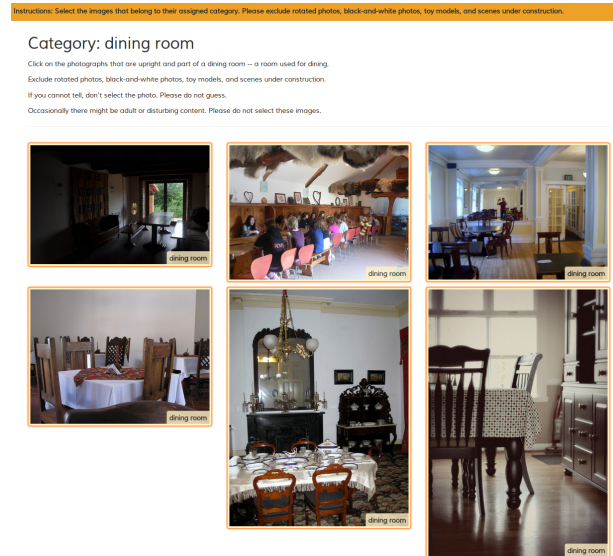


Figure 1: Interface for filtering images that do not match their scene label.

Instructions. Users are instructed to click on pictures that match a specific category label, while excluding:

- Rotated photos (an overhead view is acceptable)
- Black-and-white or sepia-tone photos
- Photos with special effects or superimposed text
- Very blurry photos
- Rooms under construction
- Toy models of a scene
- Naked people or sexual poses
- Pictures of mostly people, where the scene is not clearly visible.

Interface. Users are presented with a grid of 50 images, each annotated with the Flickr search query (e.g. “living room”, “kitchen”, etc.). The photos are grouped by category, and then sorted by aspect ratio to make the grid align nicely. When a user selects an image that matches its query, the label changes from “not living room” to “living room”.

Feedback. We found that workers were very reliable at this task, and many appreciated the large buttons (“everything is very well done, easy to click (thank you for no bubbles!) Great job with the interface, it is very easy to work with!”).

4 Flag images with improper white balance



Figure 2: Interface for clicking on white points in images. Each HIT contains a batch of 10 to 15 images.

Instructions. Users are instructed “For this task you will click on three white or gray things in the image. Click on items that should appear white or gray if placed under white lighting. Avoid extremely bright or oversaturated spots.” For guidance, users are shown 9 good examples and 2 bad examples.

Interface.

- If a user clicks on an oversaturated point (\geq RGB 253, 253, 253), they are interrupted with the message “Please avoid oversaturated spots. Click on the Instructions button for more information.” and the point is rejected. The user must click elsewhere.
- The user may click “There are no more white things”, at which point they are prompted with “You have selected 0 white spot(s). Are you sure that there are no more white or almost-white parts of image?”.
- If the user clicks 70 pixels near an existing point, the existing point is deleted and no new point is added
- Users can view the instructions inline at any time with a modal dialogue.

Feedback. Users generally liked the task and understood all of its aspects. While many asked for more pay and some expressed generic confusion, the general consensus was “I think it’s a well designed layout, nothing comes to my mind that can improve it.”

5 Material segmentation

Instructions. Users are instructed “Your task is to draw polygons around regions that have a single type of material or texture.” Users are shown many examples of good and bad segmentations, many common errors are discussed, and all interface controls are described in detail.

Interface. Before designing our interface we tried tools published by other authors (e.g., LabelMe). We opted for a more complex interface to encourage better segmentations. Our feedback has been very positive, and many users thanked us after we added each feature.

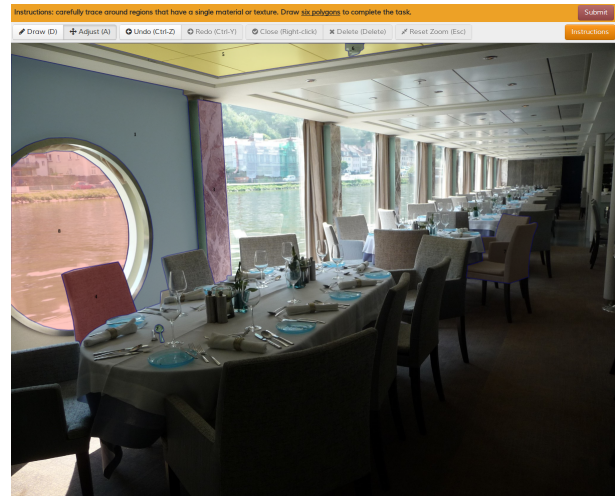


Figure 3: Interface for material segmentation. Each HIT displays one photo.

Below are descriptions of interface features:

- The interface has two modes: draw new polygons and adjust existing ones.
- Users may left-click along the border.
- Right-click will close the polygon.
- In adjust mode, the window zooms into the shape and the user may drag the vertices freely or delete the shape entirely.
- We store a full undo/redo stack for all actions and log every action with timestamps. The log is accurate enough to replay all actions (including invalid ones) at the original speed. In future work we may analyze or visualize these logs.
- We prevent polygons from self-intersecting during both the drawing and adjusting stages. A red line flashes to highlight intersections, and then the polygon is reverted to a non-intersecting state. On every third attempt to make a polygon self-intersect, the user is shown instructions explaining the problem.
- The user may zoom in and out with the scroll wheel.
- The user may pan around the image by holding space.
- If the user clicks near the edge, the image pans automatically.
- We show inline instructions with a modal dialogue for easy access.
- We allow nested and intersecting shapes (though not self-intersecting).

Feedback.

- Users seem to enjoy it, but generally requested more pay:
 - “It is fun and relaxing actually. I like focusing on the items and ensuring that the polygon is right-on the money!”
 - “It’s easy and fun. However, it is not worth time like real job.”
 - “I like it, but it should pay a few cents more!”

- Users seem to enjoy the zoom functionality and often labeled tiny objects.
- There was wide variety in the quality, though some were surprisingly good.
- There many submissions of individual bricks and tiles – either they were being time-efficient, or misunderstood “single material or texture.”
- Some people did *object* segmentation instead of material segmentation. The boundaries they provided were quite accurate, but contained multiple materials. For these shapes, some of the boundaries are correct, but not all of them. If we can identify these shapes, it would be possible to subdivide them by sending them through the pipeline again.

5.1 Algorithm for segmentation intersections

Objects often contain large regions of one material, with small regions of a second material (e.g., a door with a handle, a shower with a drain, a toilet with a metal handle). To facilitate the labeling of these surfaces, users can provide the boundary of the outer shape, and the boundary of the inner shapes.

We process these with a simple algorithm.

Input: a list of poly-lines from the labeler. For robustness, we do not require the input to be non-self-intersecting.

Output: a list of complex polygons, each polygon represented as a list of vertices, triangles, and boundary segments.

Algorithm: 1. Place all of the input line segments into both a constrained Delaunay triangulation (CDT) and a 2D arrangement (using CGAL). 2. Reject CDT triangles that are not inside the arrangement (since the CDT triangulates the convex hull of the input, thereby producing extra triangles).

3. For each pair of neighboring triangles, if they are in the same arrangement face, cluster them together. Each cluster becomes one output complex polygon. To associate output polygons with the input poly-lines, the arrangement of each input poly-line is constructed; if a cluster is entirely inside one of these poly-line arrangements, they are associated together. Unassociated polygons are discarded. If an output complex polygon is associated with more than one input poly-line, metrics such as timing are averaged together. The clustering is computed with UNION-FIND. All computations are done with exact constructions and predicates.

6 Quality votes for material segmentation

Instructions. Users are instructed “Click on the red shapes that contain a single material or texture. Reject shapes that are far from the material boundary.” Users are shown 16 good and 16 bad examples randomly selected from a collection of 300 images.

Interface.

- Users are presented with a grid of images, zoomed into the segmented region, with the shape border indicated in red. If the user hovers over the shape, the area will be highlighted; this is useful if the shape is not a simple polygon and it is unclear which pixels are part of the shape.
- Users click on the shapes that have a good boundary and contain a single material.
- If 3 of 5 users click on a shape, then it is considered “high quality”.



Figure 4: The first 8 of 32 examples on the instructions page for quality voting

Feedback.

- There are many ambiguous cases, and this showed up in the results. For example, some shapes have a beautifully detailed boundary, but is truncated because the user who drew the shape had zoomed in and forgot to zoom out. In other instances, it is unclear what constitutes a “single material”. While generating our examples, it was difficult to stay consistent. We received user feedback about the ambiguity:
 - “This is very Interesting. Some Times I Have a Confusion about the Boundary Line. Any Way I am Willing to do this kinds of hits. example: You have a Plate of pizza. One is pizza (pizza is Full Of The Plate) and other one is plate. How Can i Treat This? Is it a multiple texture or Single Texture?”
 - “a few were ambiguous - tile and grout, or insulation with a ripped lining”
 - “Maybe add a comments section to better identify ambiguous images”
 - “Examples are slightly confusing - in one case a cupboard door is marked as incorrect and another as correct without real explanation as to what the differences are”
 - “Texture is ill-defined. It is a bit unclear how to treat objects like a window or mirror. Clearly they have the same texture in real life when touched but in a picture the perceived texture is that of the surroundings. Also, what about highly patterned objects where the pattern doesn’t repeat itself much? Visually it seem as if the texture changes.”

Qualifications. As we were running the material segmentation stage, we noticed that a few users produce exceptionally detailed segmentations, with an accuracy higher than the output of the voting step (described above). After collecting 25,000 segmentations, we restricted the task to the best 26 workers (out of 530,

using MTurk qualifications) and removed the voting step. This both doubled the average detail from 11.6 to 20.3 vertices and reduced our total effective cost from \$0.035 to \$0.025/shape (including bonuses). Costs were reduced since we were no longer paying for voting or for shapes that would inevitably be rejected in later pipeline stages.

7 Planarity labeling

Examples (side view)

To explain what "planar" means, consider these examples. They are side views of objects.



Examples

Positive examples



Negative examples



Figure 5: The first 10 planarity voting examples. 32 examples are shown to users.

Instructions. Users are instructed “You will be shown a collection of images with objects outlined in red. Click on the objects that are planar – regions with a *single* flat surface. Planar shapes can have some variation, such as patterns on wood, or grooves between bricks, or fuzz on a carpet. For example, a brick or tile wall is still considered planar. However, two very different planar surfaces (such as two perpendicular walls in the same red shape) do not count and should be rejected.” Since users struggled with the idea of planarity, we focused on common mistakes for our negative examples.

Interface. Users were shown a grid of images arranged similar to the examples. They were instructed to click on the planar segmentations.

Feedback.

- People seemed to like this task and find it easy; almost all feedback was similar to “This task is very interesting and easy.”
- Users were interested in the task,
 - “it is interesting for the potential benefits of such a database”
 - “This task is actually quite interesting, if one thinks about the numerous applications of such a surfaces database. First thing I would think about would be game creation, as in a sim world.”
- People also found it to be well defined: “It’s simple to do and make us to think of which is the correct image. I understood all the parts in this task. All things are clearly defined.”
- We did not implement zoom, but feedback suggested it would be helpful: “Some pictures were difficult to distinguish whether it was a flat surface even when I magnified the picture.”
- We included a “I can’t tell” choice, which we count as “No”, in response to the feedback “There should be a choice that unable to tell if it is a flat surface. Some pictures are not very clear. Also hard to tell if the depicted area includes handles or curved edges like in a picture frame.”

8 Material names

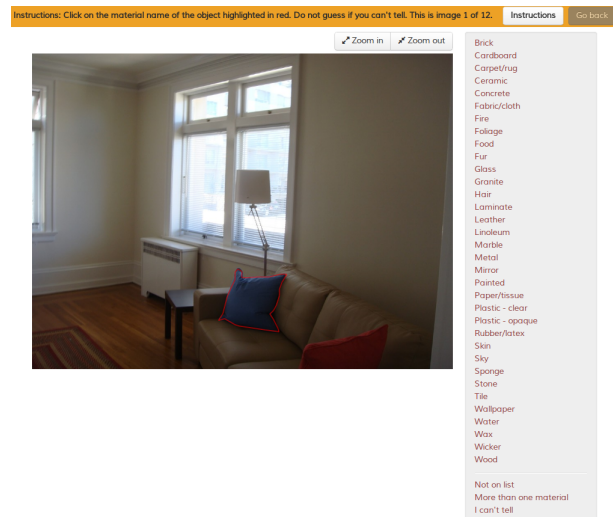


Figure 6: Material naming interface. Each HIT displays 40 segmentations in context, one at a time.

Instructions. Users are instructed “Click on the material of the object highlighted in red. Do not guess if you can’t tell.”

Interface. To collect a material label, we show the photo with the region outlined in red. An animated rectangle zooms into the region to direct the user’s attention (it begins by enclosing the entire window and decreases in size). The region then flashes white. Before we added these animations many users complained that they had trouble finding the surface. Once a user clicks on a label, the next image is presented. Users may go back if they make a mistake. Users may also zoom in/out and drag the image around.

Material definitions

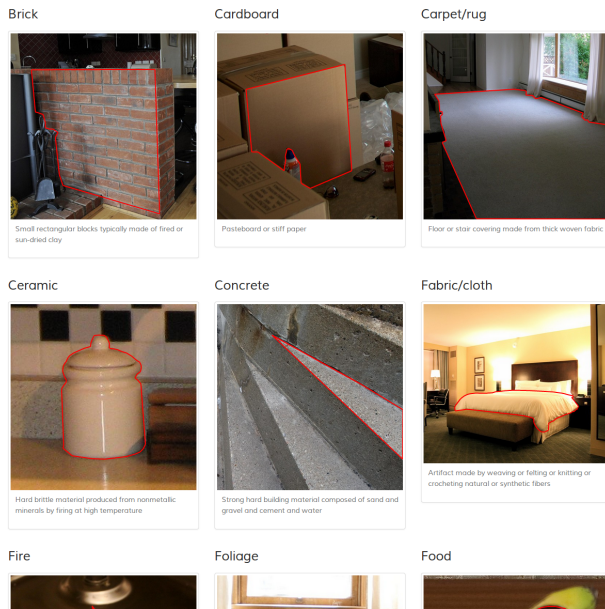


Figure 7: Examples for each material name. Only 6 are shown here; users are shown one example for every material.

Name list. The list of materials was a combination of the Flickr Materials Database (fabric, glass, leather, metal, paper, plastic, stone, water, And wood), a list we derived from our pilot study (brick, ceramic, fabric, food, fruit, glass, laminate, leather, metal, paper, plant, plastic, stone, water, and wood), and from browsing the dataset and writing down common materials. As the task progressed, we added materials to the list and had some materials re-labeled. Our final list was restricted to 33 names so that it fit within one screen. Users are presented with a list of 33 materials, plus “Not on list”, “More than one material”, and “I can’t tell”. As we collected semantic labels, we found it useful to transform some categories (e.g. rename “Paper” to “Paper/tissue”, and “Carpet” to “Carpet/rug”) to cover some confusions and common synonyms.

The full list of materials is: *Wood, Painted, Fabric/cloth, Glass, Metal, Tile, Carpet/rug, Plastic - opaque, Ceramic, Paper/tissue, Granite, Food, Leather, Concrete, Marble, Laminate, Brick, Plastic - clear, Mirror, Cardboard, Stone, Wallpaper, Wicker, Skin, Hair, Fur, Foliage, Wax, Rubber/latex, Sky, Sponge, Linoleum, Fire, Water*

Feedback. Workers greatly appreciated the example image for each material. Otherwise, there was no strong opinion from workers: “easy”, “Its cool”, “interesting”.

9 Object names

Instructions. Users are instructed “Click on the name for the common English name of the object highlighted in red. Do not guess if you can’t tell.”

Interface. The interface is otherwise the same as material naming. Due to the large number of items, we did not show a canonical example for each object. This did not seem to affect worker accuracy (whereas showing examples helped significantly for materials).

Name list. For each material, we select a subset of the object names that are relevant. The full list of object names is: *Animal, Bag, Basket, Bathtub, Bed, Blanket, Window blinds, Book/magazine, Bookcase, Bottle/jug, Bowl, Bread/toast, Cabinet, Cake, Can, Candle, Ceiling, Chair, Cheese, Clothing, Column, Container, Cookie, Cup, Curtain, Cutlery/utensils, Cutting board, Desk, Dishwasher, Documents, Door/doorway, Dough, Drawer, Dresser/armoire, Egg, Electronics, Envelope, Faucet, Floor, Food, Fruit, Glasses/sunglasses, Handle, Hat, Jar, Kettle/teapot, Label, Light/lamp, Mantelpiece, Map, Mattress, Meat/fish, Menu, Microwave, Money, Muffin/cupcake, Mug, Napkin, Napkin/tissue/paper towel, Newspaper, Ornament/sculpture, Outlet/switch, Oven/stove, Painting/poster, Pasta/noodles, Person, Pie, Pillow/cushion, Plant, Plate, Present/gift, Railing, Refrigerator, Rice, Saucepan/pot/pan, Shelf, Shower, Sink, Sofa/couch/armchair, Soup, Stairs, Stool, Table, Tablecloth, Tapestry/quilt, Television stand, Television, Toilet paper, Toilet, Towel, Tree/branch, Trim, Vase, Vegetable, Wall, Window, Wine glass, Worktop/countertop.*

We expect to expand our list of names in future work. The list was built by studying the shapes that failed to receive a label (either it was marked “Not on list” or it had high disagreement). When we expanded the list, we put the problematic shapes back into the pipeline. Our preliminary studies found that freeform inputs lead to inconsistent and misspelled answers. However, we may consider allowing freeform inputs for objects that did not receive an object label.

10 Rectification

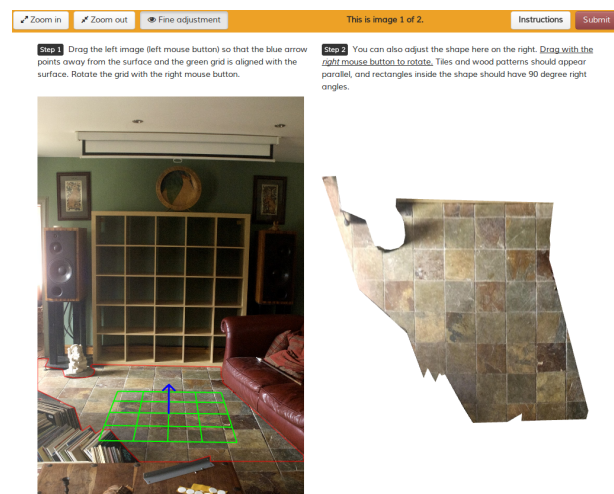


Figure 8: The rectification interface. This figure shows a successfully completed task where the perspective grid appears to lie flat against the surface (left half) and the texture is correctly rectified (right half).

Instructions. Workers were given detailed instructions and many examples (Figure 9). “**Step 1:** Drag the left image (left mouse button) so that the blue arrow points away from the surface and the green grid is aligned with the surface. Rotate the grid with the right mouse button. **Step 2:** You can also adjust the shape here on the right. Drag with the right mouse button to rotate. Tiles and wood patterns should appear parallel, and rectangles inside the shape should have 90 degree right angles.”



Figure 9: Rectification examples. Users are shown 32 examples in total.

Interface. As shown in Figure 8, users can drag the 3D perspective grid to adjust the surface normal, or they may drag the rectified result directly. Dragging in the left half adjusts the angles in camera space, while dragging in the right half adjusts the angles in object space. The result is rectified in real-time using a WebGL scene, with the desired transform encoded as in the camera projection matrix. To render the rectified shape, the 2D triangulated mesh is used for geometry, and the bounding box of the shape is used as a texture.

11 Rectification quality

Instructions. For this task, we instructed users: “For this task, *rectified* means:

- the blue arrow points away from the surface,
- the green grid is parallel to edges in the surface, and
- tile, brick, and wood patterns should be *perfectly* aligned and the same size throughout the image on the right.”

In order to get workers to properly select good rectifications from bad ones, it was necessary to show them negative examples that were only slightly incorrect. Otherwise, they were not picky enough and would select almost every item as a good rectification.

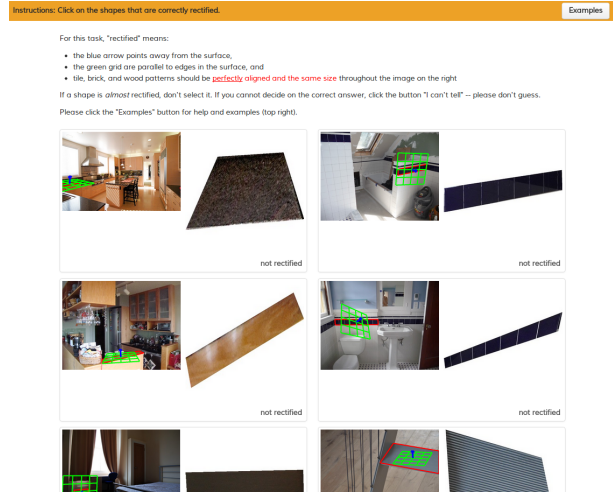


Figure 10: Interface for filtering bad surface normals. Each HIT is 40 rectifications.

We use the same example set for both this task and the previous one (rectification).

Interface. The interface (Figure 10) is the same as quality filtering for material segmentations. Users select the rectifications that are correct from a grid of 40 examples. Users can also select “I can’t tell” which counts as a negative vote (since we only want surfaces that users can positively determine are correct).

Feedback. Feedback was split between total understanding and total misunderstanding. We speculate that since the task is more complex, the users who struggled with the task were those which were not fluent in English.

12 Appearance matching

Instructions. Users are instructed to adjust the appearance of the blob until it matches the target.

Interface. The blob is rendered in realtime using a WebGL scene. The rendering method is described in the main paper.

Feedback. The general consensus was “It’s interesting. It can be a challenge to get the adjustment close.” Workers seemed to have a consistent level of quality. Some workers had a bias to consistently over-estimate the contrast, submitting blobs that did not appear to match at all. Others would consistently set the distinctness of image (d) to the middle setting. A few seemed to really understand the task and consistently submitted good matches. We targeted these users for bonuses.

13 Color and gloss quality

Interface. The interface (Figures 12 and 13) is the same as the other quality interfaces. First, color is checked, and those that pass color are checked for gloss. Since the input to the gloss check are all blobs with matching color, users are not distracted by considering multiple aspects at once.

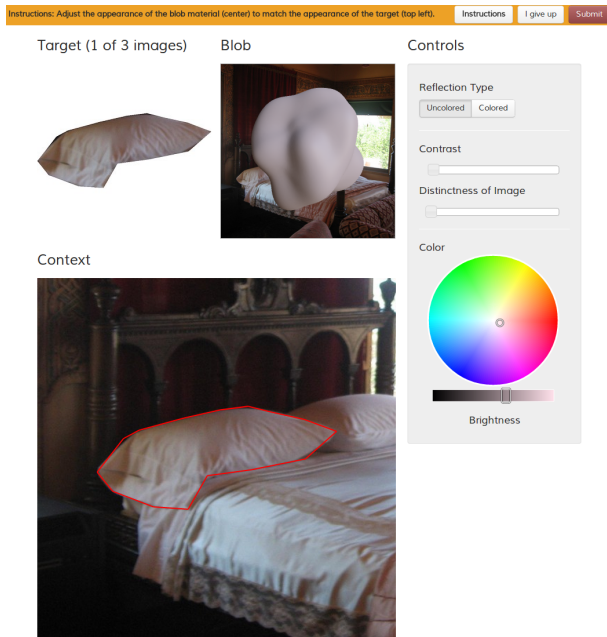


Figure 11: Appearance matching interface. Both photographic context and shape are provided.

Feedback. Workers seemed to like this task; some called it “cute”, others said it was “useful”. A few workers noted “Sometimes matching the colors is difficult especially if the pattern is varied.” Generally the workers were happy with the pay for this task.

14 Feedback given to MTurk workers

When workers submit results, we estimate a *badness* score for each assignment. For binary tasks, we use the fraction of disagreement with the CUBAM result; for naming tasks, we use disagreement with the consensus for each shape; and for segmentation, we use the fraction of shapes voted as high quality. If the badness is greater than half, then we flag the HIT (except for segmentation, where it must be ≥ 0.9). Since automated rejections can lead to unfair decisions (thus chasing away the best workers), we manually review all potential rejections. This ensures we maintain a good relationship with our workers.

If the badness score is less than half, we approve the HIT and provide a message ranging from “All of your submission was bad. We approved the HIT since we did not want to hurt your qualification score” to “Perfect submission! Thank you” depending on the score. We found that workers greatly appreciated this feedback. Many sent us emails thanking us, and others used our feedback form. One worker wrote “it’s difficult but i appreciate your positive feedback when you approve/reject the HITs, so i’m motivated to please you!”

For segmentations, we set the badness threshold to 0.9, since we noticed that workers either consistently submitted completely invalid results (badness near 1.0) or perfect results (badness near 0.0). When we started the tasks, we rejected roughly half of the HITs. After the task had been running for over a week, workers responded to this feedback and we were only rejecting a few percentage of HITs. With this system, both parties benefit—workers do not get paid for rejected HITs, and each rejection reduces the number of HITs they qualify. Eventually, we switched to using qualifications (for segmentation only) and rejected none of the segmentation HITs since every submission was near-perfect.

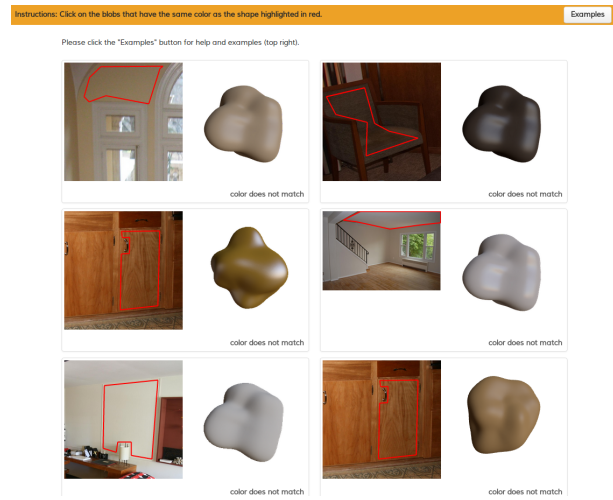


Figure 12: Interface for filtering bad color matches

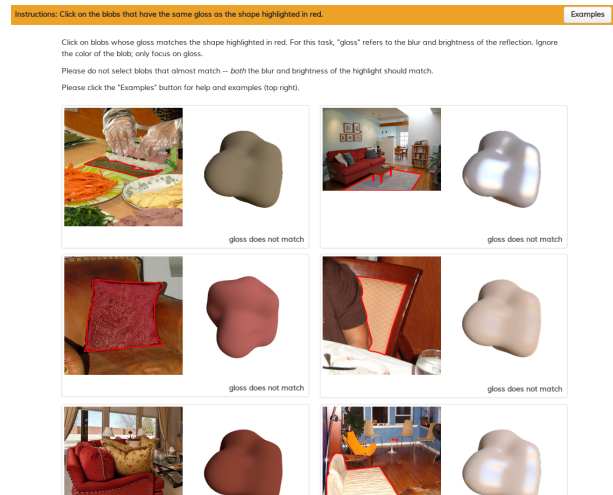


Figure 13: Interface for filtering bad gloss matches

15 Tools and implementation notes

- The backend is mostly Python running Django.
- The front end is written in a mix of Django templates, CoffeeScript, and Less. For each HTTP request, the Django template is compiled into HTML, the CoffeeScript is compiled into JavaScript, and Less is compiled into CSS. To improve performance, it is all cached in memcached.
- To display large amounts of data, results are paginated dynamically; scripts in the browser automatically fetch and insert new rows of data as the user scrolls.
- Asynchronous processing uses Celery and RabbitMQ. When MTurk workers submit data, their POST data is put on a queue so the worker can immediately start the next task. The server then asynchronously processes the data.
- The MTurk annotation pipeline is set up as a producer-consumer chain. Experiments may produce results asynchronously in parallel, and “consuming” results are serialized into a single queue. As results are produced from one experiment, they are filtered and re-grouped for the next experi-

treilmann trekbuje trevgrant trevira trevorandmarjee trevorpatt trevorschwelnius
trey333 triaciawang trick_ tripliferivredw tripliyew tripmeter9 trippchicago tripu
trishhhhh tristanft tristanlontr tritoliti trix_smith trixoux troynt troublemakersinc
trove trucolorsflly truebritishmettal truusbobajantow trvance trydberg tsakshaug
tsand tschaut tstadler tsuki_chama ttcopley ttoocland ttrueman tryggve ttstam
ttsy0 tubagooba tubby tubbyjy tucoman tuhochi tuckster tudor tulanesaily
tunnelbug tupwandens turbojoie turbormz2 turbulentflow turgeon turoczy turgy
turukhtan tusela tuttotutto tuursenzo tvancort tveng tvot twak tweedeedesigns
tweenina twenty_questions twentysixcats twic twiceipix twiga_269 twinlaives
two-wrongs twobears2 twodolla twohungrydues twonickels twoes twoersg tooof
tyfn tyle_r tylerekaraszewski tylerkathe tymesynk tyng typhen tyberind typicalcuser
tyrian123 tzachernak-pics tztkk u0304380 u07ch uberculture uberomooole uberzombie
ujbrayj02 ucdaviscoo usec usfv uagcommunications uagardener ugboyj utdragerij
ukrajien ukanda ulfbodin ultimatalibrarian ultravod umami88 umdent umdnj_sn
umdrums ume-y umhealthsystem ummella un_col_a unavoidablegrain unco-cfc-usfk
unleboatshoes unconsciousstrees uncolobol uncorrectedproofs underdutchskies
unserore unfoldedorigami uniondocs unknowndomain unloveable unstoppleddrew
unreshot unwiredben uordenver upsand upornoz upyoureger urban_dacia urban_lenny
urbanduck urbanlatinfemale urbanmappper urbanmixer urbanwide urberplo urbrnthkr
uriba urish usamajor us_ambassador_gutman usace-kcd usachicago usauf87
usag-yongsan usagpak usagumphreys usagvencia usaid_images usairforce usarak
usarmyafrika usasq usdagov usnameixists usfsmidwest usfssoutheast ush usnavy
uspyhe utahstatelibrary utapistm utt73 uuzinger uvuphotos uwpoabmissions
uwresnet uxud victory_ls_ism6 v63 vacationtime vadr vadin_trochinsky
vagabondblogger vagabondtravels vanguardao vagueonthehow vaingloro valakirka
valeriebb valerioevo valkiriehl16 valt98 vampiressl44 vancouverconvention
vanderlin vandypants vanessaberry vanhoosear vanschijnjeljan variationblog
varlasdrive varrgnuht vaneska vastateparksstaff vasvalch vauvau vax-o-matic
vecker vdm vev_smith veganfeast veganheathen vegansoldier vegantraveler veilsto
vek veikr0 venana veni venosdale ventanuzd venuemisce vermicious_knid vero-b
versageke vertigoacid vestman vfwolvr vgm8383 vgnuda vhwone vliagem secreta
vialbost vibrant_art vicarsh vickyy victoriavictorian victoriano vidya365 view vike
villazeros ville-arles vilseksosgen vlineone vinsense vintagedeep
vintagefindings vinzcha vioffidder vipnyc virabius virginied virginemerry
virionny virovets virtualcounrney virtually_supine virtualsurgar visionsvicky
visionshare visitidveon visitfingerlakes visitflanders visittallinn vissago
vistamonster vitamindave vitodens vitorpamplona vitpnm viuocr vividcorvid vib_x
vk2gmk vkan vkreay vmiramontes voght volk vonderaivsuals vonlohmann
voodooangelmg vorn vortech vortepine vox_ofk vpx vr vsank vserys vsmituch
vtcarter vxla vyelevich w00kie w5fny wabson wackelljmooster wackybadger
wadebrooks wadecouch wader wadetreaskis waffer wagneriteam waithere waiferx
walkin wakest wakxy waldepndon walker_ep walkslide wallyhartshorn walschots
walshetta waltermenzies waltervargas walstoenburner wamble wandererwoman
wandrus wangkai wangxu94 warrenintweeds warrenjackson warthog9 wasav
washingtombc washuugenius wasms watertownsurfer wattedas watz wavy1
wayneandwax wd4k wbavi wbeem wbarrison wcamliin wdcomdents wdecora wdobarber
wearechapterone weasel-on-wheels webdiva webmik weebum weedrummy weefz
weekilyd weidnerapartmenthomes weinelt weirdo513 wellohorid welovethesky
wendolf wendles56 wendy wendycopley wendymehndi wendyness wonderobotika wengs
wendolton wesker1972 wesleyt wessexarchaeology west_point weston1 westsidelaura
wetchman wetfeet2000 wetwebwork wfiupublicradio wfmu wfuv wfyurasco wginflie
whalt whartz whatcouldgowrong whatdavesees whatsleft whatnot whattappics wheany
wheatfields wheatland wheaton wheavil whispervolf white_wolf whitecrafts
whitecatsg whitnuld whitetbread whoatsaimz whoisthatfreakwiththecamera whsieh78
whyamkienna whybesubtle whyssnomen wiccked wickenden wickham widomirama widvey
wien-vienna wikidave wikipix wilbanks wildchild wildflowersflorida wildtexas
wildtraveldeals wilhelmja willi willbakker willemvanbergen willia4
william-and-mary william_v williamw williamd williamh williamhook william5
williamspictures williumwillin willowherb wilsonb wilwheaton willwright
wind-river windsordi wintrhawk wiredtuch wisny wisley wistreise witemikel015
win-an-ey wihassociates wiwin_wr wjw wjkjoks wjlonnie wa kwharmon wcutler
wlo1 wisclence wm_archiv wm_gary_warren wmaacphall wmshc_kwitayor wneuhiesel
wolfholdr woick woicjeh-przybylski wojohwitz wolfe-mckee wolfgangstaad
wolfhunter wolman-k wolframbaner wolfsavard wolfworld wonderferret wonderlane
wonderwebby wonko woodpainter woods-kimber woodsm woodyellen woof_pup woowie2010
wordriden worldofjan worldoflford worldfordoddy worldsworstophographerintheworld
worldtoblate worldworldworld woscholar wounded woude wramedia wrestlingentropy
wrote wselman wtrauss wudrich wumbus wundoroo wuniatuph wunkaiwang wurz wv
wvagent wwarby wwf-France_footage wward00 wwanconxs179com wwbeber wwkanlorg
wworks wwupertal wxmwm wy_jackrabbit wyandnavpoortvilt wyner3 wyrmworld
xycan wysz x-o-ph xlbrett xabbu xabiericid xadoom xaf xamichee xand83 xandra_lee
xavitalleda xcaverx keelz xerones xerostomia xesc xiaming xiaozhou1 xispa
xixidu xiliber xmanst1 xmitter xmracks xopherbrown xoque xoxoryan xraixs xxxx
xshamesthextronyg xt0ph3r xtl xuecuro xwl xkroberger xystance ya3hs3 yacomink
yaffmedia yaill yakobusan yakstargher yama2k yamahawatercraft yandah yandle
yannic yarboroughs yarharqoat yarmtheshore yashima yashna13 yasuihoroa yauhin
yazuu ybla ybnormalm ydshu yellowbookltd yellowskyphotography yelnoo yenna
yenny yeozatzip yersinia yevgene yewenyl yl yimmy149 yinghai83 yisongyue
ylakeland ylegrand ynthesiser yodelanecdotal yodster yogma yoghanes yohie yomie
yopizza yorgda yoryi yosnowden yungdoou your_wht_knight yourbaterand yourdon
yourworldmissionaries youssefabdelkhal youthradio yrrek yuan2003 yugen
yukoachatulapoly yule yumeto yumum yuyang226 yveshanouille yvoictta yvonnexlai
z-2wo z303 zabbid zak zacharyparadis zachahle zachinglis zachklein zachlur
zachtwos zachno zachn zagrev zala zach zamboag zamkov zanador zanastardust zandperl
zaneholingsworth zaneselvans zanic zanthia zapowbang zarabee zarrion11
zazasvq zbtwells zcopley ze_valdi zebe1 zebranete zebtrn zeeuwsebibliotheek
zeimke zemzina zen zencat7 zendritic zenobia_joy zepfman zersoian zetgem
zhanameloc zigazou76 zigdon zitona zivkovic zixi zmackid zmtakmo zoer
zoerochelle zoetnet zokuga zolk zombiete zormerztoom zoomar zorbs zordroyd zota
zpeckler zric zsvna zsolitika zsrilibrary ztephen zww zyilan zykomia zscaspe
zzazazz -erin -jvl -wichid -will- -Olisteven 0321recon 050079 0x 1-25 sbct
1-6-scale-doll-clothes 99893469NO 90943070NO 100047629NO 100681738NO 100sf
10164913NO 10165212NO 10205167NO 1024 10248895NO 10348212NO 107
10370393NO 10413034NO 10557450NO 10558467NO 10623456NO 10727620NO 10
10789705NO 10865676NO 11152520NO 11285577NO 11387874NO 11401127NO 11
11401580NO 11436660NO 11452088NO 11492088NO 11561957NO 11628368NO 107
11632301NO 11638547NO 11716680NO 11777792NO 11901158NO 11921146NO 10
12019105NO 12265657NO 12313953NO 123744cd 12394349NO 12433001
12482365NO 12495774NO 12535240NO 12636945NO 12657713NO 12655485NO 12
12663660NO 12738795NO 12784349NO 12866649NO 129440906NO 12949421NO 12
12986948NO 13105121NO 13218823NO 13149474NO 13151038NO 1337H4x0r
13382424NO 13385504NO 13422016NO 13706945NO 13712921NO 13825348NO 13
13942016NO 13960158NO 13961ngdms 14092627NO 14270634NO 14589121NO 10
14646075NO 14657061NO 14659638NO 14667107NO 14723335NO 14863785NO 13
148degres 15052678NO 15132846NO 15134271NO 15157516NO 15216811NO 10
15421875NO 15427016NO 15481483NO 15609463NO 15708236NO 15985028NO 10
16307858NO 16315844NO 16330500NO 16339612NO 16497759NO 16588248NO 10
16725630NO 16734870NO 16854395NO 16880572NO 16986595NO 17067947NO 10
17195501NO 17421847NO 17489999NO 17548547NO 17755777NO 178047758NO 10
17989497NO 18091975NO 18203311NO 18378305NO 18415326NO 18452723NO 10