Segmentasi, Pelacakan, dan Pewarnaan Waktu Nyata dari Dinding Menggunakan iOS

David Liu
jurusan Teknik Elektro
Universitas Stanford
davliu@stanford.edu

Jeff Piersol
jurusan Teknik Elektro
Universitas Stanford
jpiersol@stanford.edu

Serena Yeung
jurusan Teknik Elektro
Universitas Stanford
syyeung@stanford.edu

***Abstrak*— Dalam karya ini kami menerapkan algoritma pemrosesan gambar pada platform iOS yang mengubah segmen dinding yang dipilih pengguna menjadi warna yang berbeda secara real-time. Secara khusus, kami mendemonstrasikan algoritme yang menyegmentasikan, melacak, dan secara realistis mengubah warna segmen dinding pada kecepatan bingkai video. Tantangan signifikan dihadapi dalam mempercepat waktu pemrosesan per frame ke tingkat yang dapat diterima, dan akselerasi GPU perlu digunakan untuk beberapa tahap pipeline untuk mencapai kinerja yang diinginkan. Hasilnya menunjukkan prototipe sukses transformasi warna dinding real-time, serta peluang untuk pekerjaan lebih lanjut untuk meningkatkan kecepatan dan akurasi segmentasi.**

I. PENDAHULUAN

Lukisan dinding adalah proyek renovasi rumah umum yang biasanya melibatkan pemilihan warna cat dari ratusan kartu cat kecil. Mungkin sulit untuk membayangkan seperti apa hasil akhirnya, dan untuk memilih di antara begitu banyak warna yang agak berbeda. Biaya pengecatan dinding yang mahal juga mencegah trial-and-error, dan meningkatkan tekanan untuk memilih warna yang tepat pada percobaan pertama. Aplikasi seluler yang dapat menunjukkan kepada pengguna seperti apa tampilan dinding setelah mengecat, terutama dengan pengalaman real-time yang alami, dapat menjadi alat yang berharga untuk membantu orang memilih warna yang tepat untuk mengecat dinding mereka.

Dalam karya ini kami mendemonstrasikan aplikasi seluler yang dikembangkan pada platform iOS yang cocok untuk aplikasi semacam itu. Aplikasi kami memanfaatkan aliran kamera langsung dari perangkat iOS untuk melakukan tiga tugas utama: (1) membagi wilayah dinding berdasarkan titik benih yang ditentukan pengguna; (2) melacak wilayah dinding saat kamera bergerak; dan (3) mengubah warna dinding menjadi warna yang dipilih pengguna. Semua ini dilakukan di bingkai video tarif sehingga saat pengguna melihat melalui kamera perangkat iOS, pengguna melihat dinding seolah-olah telah dicat dengan warna berbeda.

A. Pekerjaan Sebelumnya dan Terkait

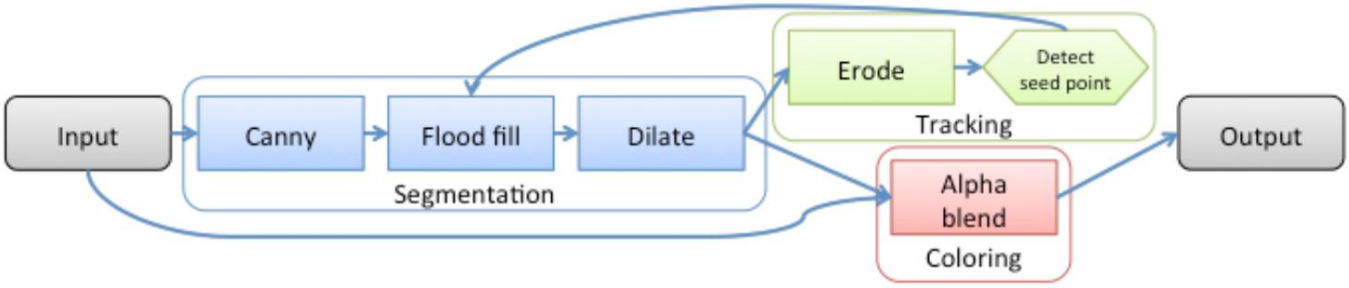
1) Segmentasi: Segmentasi berbasis warna dan tepi paling sesuai untuk dinding karena permukaannya yang biasanya besar, datar, dan berwarna tunggal. Ada banyak cara untuk melakukan deteksi tepi, termasuk Canny, Sobel, Harris, SUSAN, dan Laplacian dari detektor tepi Gaussian. Semua detektor ini melibatkan konvolusi dengan kernel yang ditentukan diikuti oleh operasi pasca-pemrosesan seperti penekanan non-maksimum. Atau, operasi morfologi seperti erosi dan dilatasi juga telah digunakan untuk deteksi tepi [1].

2) Pelacakan: Seperti halnya deteksi tepi, ada berbagai metode yang ditetapkan untuk pelacakan, seperti estimasi menggunakan transformasi wavelet, minimisasi energi melalui pemotongan grafik dan organisasi fitur visual berbasis loxel [2] [3] [4] [5], tanpa algoritma yang optimal secara universal. Beberapa metode lebih cocok untuk lingkungan tertentu, dan selalu ada trade-off antara akurasi dan kecepatan. Misalnya, algoritma pencocokan blok dapat mencapai akurasi yang lebih tinggi melalui pencarian lengkap versus pencarian lokal dengan heuristik, tetapi ini menyebabkannya menjadi lebih mahal secara komputasi juga [6]. Selain pencocokan blok, metode pelacakan terkenal lainnya termasuk menggunakan berbagai deskriptor fitur, yang mencari korespondensi satu-ke-satu antara gambar referensi dan gambar target [7] [8], dan algoritma pengkodean video, yang mencoba menemukan vektor gerak setiap piksel berdasarkan codec pengkodean [4]. Semua metode di atas melakukan pelacakan dengan menentukan perkiraan transformasi dari gambar referensi ke gambar target melalui warping. Saat mempertimbangkan penggunaan pelacakan untuk lukisan dinding di aplikasi seluler, prioritas tertinggi adalah kecepatan dan kesederhanaan penerapan.

3) Transformasi Warna. Sulit untuk memperkirakan penampilan yang tepat dari warna cat tertentu dengan nilai reflektansi yang diketahui ke dinding, mengingat kondisi pencahayaan kamera ponsel yang tidak diketahui. Diskusi dengan Dr. Joyce Farrell telah menghasilkan ide untuk perkiraan yang berpotensi kuat, seperti mencoba memperkirakan terlebih dahulu penyesuaian pencahayaan kamera, tetapi ini ditentukan di luar cakupan proyek saat ini. Aplikasi seluler yang tersedia secara komersial yang melakukan pengecatan dinding (semua pada gambar dinding statis) tampaknya menggunakan overlay semi-transparan untuk menampilkan warna cat, tetapi ini tidak berfungsi dengan baik pada permukaan yang gelap.

II. ALGORITMA

Banyak algoritma dari literatur disimulasikan dalam MATLAB untuk menentukan algoritma yang optimal dalam hal hasil yang akurat secara konsisten dan kinerja run-time.
Pembuatan prototipe algoritma dilakukan bersamaan dengan pengembangan iOS untuk mengkarakterisasi kecepatan algoritma secara akurat pada perangkat iOS; selain itu, OpenCV dikompilasi sebagai pustaka MATLAB untuk memastikan simulasi pemrosesan gambar yang akurat pada perangkat seluler.
Lampiran C merinci algoritma lain yang dicoba sebelum sampai pada pipa algoritma akhir yang ditunjukkan pada Gambar 1.



Gambar 1. Pipa pemrosesan gambar.

A. Segmentasi

Langkah pertama dalam pipa pemrosesan gambar adalah deteksi tepi Canny. Deteksi Canny dilakukan pada versi gambar abu-abu, yang ditemukan menggunakan rumus

$$= 0{,}299 + 0{,}587 + 0{,}114 \quad . \quad (1)$$

Nilai ambang batas yang digunakan dalam panggilan ke detektor OpenCV Canny adalah 60 dan 25, yang sesuai dengan ambang batas untuk deteksi tepi kuat awal dan ambang batas untuk menghubungkan tepi yang lemah, masing-masing. Sebuah kernel Sobel ukuran 3 digunakan untuk perhitungan gradien.

Langkah kedua dalam pipa adalah pengisian banjir. Operasi pengisian banjir mengisi topeng biner dengan membentuk komponen terhubung mulai dari titik benih, di mana konektivitas ditentukan oleh perbedaan nilai warna antara piksel dalam komponen dan tetangganya. Ambang batas yang digunakan adalah 5.0 untuk ambang batas atas dan bawah, untuk ketiga saluran. Jika pengguna memilih titik benih baru, piksel tersebut digunakan sebagai titik benih untuk pengisian banjir; jika tidak, titik benih dipilih oleh algoritma pelacakan. Hasil deteksi Canny dilewatkan ke dalam operasi pengisian banjir sebagai masker input, sehingga pengisian banjir tidak melewati tepi yang terdeteksi Canny. Sebuah ilustrasi yang merinci operasi pengisian banjir ditunjukkan pada Gambar. 2.

Implementasi OpenCV dari operasi pengisian banjir mengeluarkan OR logis dari topeng asli dan daerah yang terisi, sehingga ujung-ujungnya dikurangi untuk menghasilkan topeng yang hanya memilih dinding. Topeng kemudian diwarnai dengan warna target yang ditentukan pengguna.

Operasi ketiga adalah dilatasi, yang merupakan langkah terakhir untuk segmentasi. Pelebaran dilakukan dengan elemen penataan persegi 3x3 dan mengisi lubang di topeng. Setelah pelebaran, segmentasi dinding selesai.

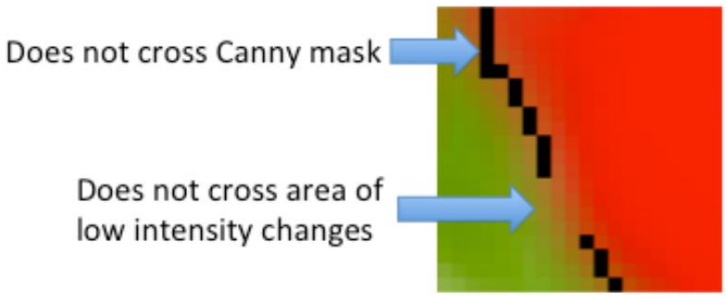
B. Pelacakan

Komponen pelacakan digunakan untuk menentukan di mana titik benih harus berada di bingkai berikutnya. Untuk menerapkan pelacakan dinding, upaya dilakukan untuk merancang algoritma paling sederhana yang akan bekerja mengingat kendala waktu nyata. Filter erosi diterapkan pada output dari prosedur segmentasi dinding, dengan elemen penataan 30x30 dengan elemen yang muncul setiap 5 baris atau kolom (dalam notasi MATLAB, SE = nol(30);
SE(1:5:akhir,1:5:akhir) = 1;). Topeng yang terkikis kemudian hanya berisi titik-titik di dalam dinding yang jauh dari tepi dinding, sehingga setiap titik di topeng yang terkikis kemungkinan akan menjadi titik di dinding pada bingkai berikutnya. Oleh karena itu, karena titik mana pun di topeng yang terkikis kemungkinan akan menjadi bagian dari dinding di bingkai berikutnya, titik pertama yang ditemukan di topeng itu digunakan sebagai titik benih untuk bingkai masukan berikutnya. Asumsi penting yang digunakan oleh algoritma pelacakan adalah bahwa pengguna tidak akan memindahkan perangkat secara tiba-tiba; jika perangkat dipindahkan dengan cepat, topeng yang terkikis dari bingkai sebelumnya mungkin tidak lagi berisi dinding, sehingga komponen pelacakan tidak akan secara akurat menemukan dinding di bingkai saat ini.

C. Transformasi Warna

Langkah terakhir dalam pipa pemrosesan gambar adalah pencampuran alfa, yang secara efektif menghasilkan kombinasi linier dari gambar asli dan topeng berwarna di wilayah topeng dan mereproduksi gambar asli di luar wilayah topeng. Pembobotan yang digunakan adalah 0,4 * gambar asli + 0,6 * overlay topeng.
Output dari alpha blending kemudian ditampilkan ke layar.

Sebuah video yang menggambarkan algoritma dapat ditemukan di [9]. Perhatikan bahwa video menunjukkan algoritme sebagai prototipe di MATLAB, bukan seperti yang diterapkan di iOS, dan algoritme video sedikit menyimpang dari algoritme akhir yang diimplementasikan.



Gambar 2. Dua mode operasi pengisian banjir.

AKU AKU AKU. OPTIMASI KINERJA

Mungkin kendala terpenting untuk pemrosesan gambar pada perangkat seluler adalah kecepatan. Dalam fase penelitian proyek ini, hampir semua operasi citra ditemukan memiliki dampak kinerja negatif yang cukup besar, yang pada dasarnya membatasi jumlah operasi yang dapat digunakan oleh program.

Salah satu cara untuk mengurangi dampak kinerja adalah dengan menggunakan perangkat keras yang mampu melakukan operasi paralel pada data, dan perangkat iOS memiliki GPU untuk tujuan ini. Antarmuka perangkat lunak ke GPU disediakan dalam bentuk OpenGL ES, yang merupakan API rendering grafis untuk digunakan pada perangkat seluler [10]. Meskipun OpenGL terutama ditujukan untuk merender grafik 3D, shader fragmennya memungkinkan operasi per piksel dijalankan secara paralel pada data, yang telah terbukti sangat berguna untuk operasi pemrosesan gambar. Operasi yang menerapkan algoritme yang sama untuk setiap piksel sangat diuntungkan dari pemrosesan GPU yang diparalelkan, sebagaimana dibuktikan dalam kolom “filter dilatasi 3x3” pada Tabel 1. Dalam proyek ini, kedua operasi morfologis dilakukan menggunakan GPU, serta operasi pencampuran alfa , meningkatkan kecepatan bingkai dari di bawah 1 bingkai per detik menjadi sekitar 5 bingkai per detik di iPad generasi ke-3.

Pada Tabel 1, tampilan frame rate untuk menerapkan dua operasi yang berbeda ditampilkan: satu baris menunjukkan aplikasi khas dari operasi, yang kedua menunjukkan operasi saat dijalankan pada GPU, dan baris ketiga menunjukkan frame rate tanpa menerapkan operasi yang ditentukan . Untuk mengumpulkan data ini, program dijalankan menggunakan algoritme yang dijelaskan di bagian 3, sambil memodifikasi penerapan filter dilatasi atau filter pencampuran alfa dengan menerapkan filter menggunakan OpenCV atau dengan menghapus operasi sepenuhnya.

Salah satu peluang untuk peningkatan kinerja di masa mendatang adalah menerapkan semua operasi pemrosesan gambar dengan OpenGL, baik untuk memanfaatkan operasi paralel maupun untuk mengurangi biaya pertukaran data antara CPU dan GPU.
Tabel 2 menunjukkan perkiraan durasi semua operasi pemrosesan gambar untuk proyek, di mana operasi 1-7 semuanya dilakukan pada CPU. Jika operasi ini dilakukan pada GPU dan mencapai percepatan faktor 2, kecepatan bingkai akan sangat diuntungkan.

Kerangka kerja GPUImage digunakan dalam proyek ini untuk menyederhanakan komunikasi dengan GPU [11]. Kerangka kerja ini menyediakan filter pencampuran alfa yang digunakan di ujung pipa dan juga menyediakan template yang berguna untuk membuat filter erosi dan dilatasi.

Tabel 1. Frame rate untuk operator yang berbeda (dalam FPS).

	pelebaran 3x3 Saring	Pencampuran alfa
OpenCV	2.37	3.2
OpenGL (GPU dipercepat)	3.5	3.5
Tanpa operasi	3.5	6.0

IV. HASIL

Algoritme yang dijelaskan di bagian metode diuji pada MATLAB dan iOS. Hasilnya dievaluasi dalam hal segmentasi, pelacakan, dan pewarnaan.

Secara umum, implementasi tampaknya berkinerja cukup baik di sebagian besar keadaan, dan pada kecepatan yang cukup cepat untuk digunakan secara real time, seperti yang dibahas di bagian Pengoptimalan Kinerja.

Ada kasus-kasus tertentu di mana implementasinya terlihat menunjukkan keterbatasan, dan ini juga akan dibahas di bawah.

A. Segmentasi

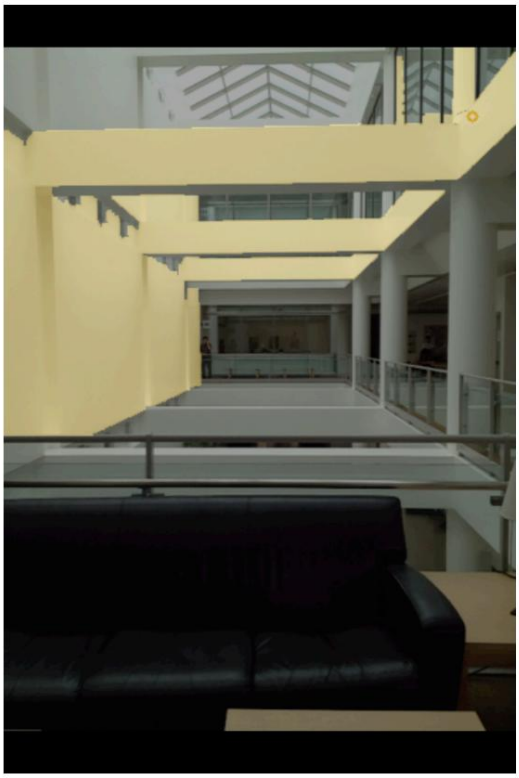
Gambar 3 menunjukkan beberapa tangkapan layar dari uji coba aplikasi seluler dalam skenario yang berbeda. Gambar 3a diambil di ruang makan, dan menunjukkan bahwa algoritme mampu secara akurat membagi garis kursi dan meja, serta batas dinding non-standar di area dapur tanpa bocor ke dapur. Gambar. 3b dan 3c menunjukkan segmentasi lebih lanjut di sekitar bingkai foto, kursi dan jendela. Gambar 3d menunjukkan pemandangan yang lebih kompleks, di mana algoritme masih berhasil melakukan segmentasi di sekitar jendela, monitor, pengaturan rak buku yang agak rumit, dan bayangan. Pada adegan yang jauh lebih kompleks, seperti yang ditunjukkan pada Gambar. 4, algoritma mulai berjuang; namun pemandangan seperti itu akan membingungkan bahkan bagi mata manusia untuk membedakan apa yang seharusnya merupakan segmen dinding.

Salah satu kelemahan dari algoritme adalah bahwa ia tidak bekerja secara reproduktif di sudut-sudut dinding: ia dapat berkedip dan terkadang mengecualikan langit-langit (Gbr. 3a) sementara terkadang tidak (Gbr. 3d). Ini karena sulitnya memilih sudut di antara bayangan di mana tiga dinding putih berpotongan. Bayangan yang kuat juga dapat secara tidak terduga memotong segmen dinding (Gbr 5). Salah satu perbaikan potensial adalah menggunakan koherensi temporal untuk membuat topeng rata-rata menggunakan beberapa bingkai, meskipun ini mungkin datang dengan kelemahan tepi yang tidak tajam secara konsisten dalam segmentasi. Perbaikan potensial lainnya adalah memaksakan lebih kuat

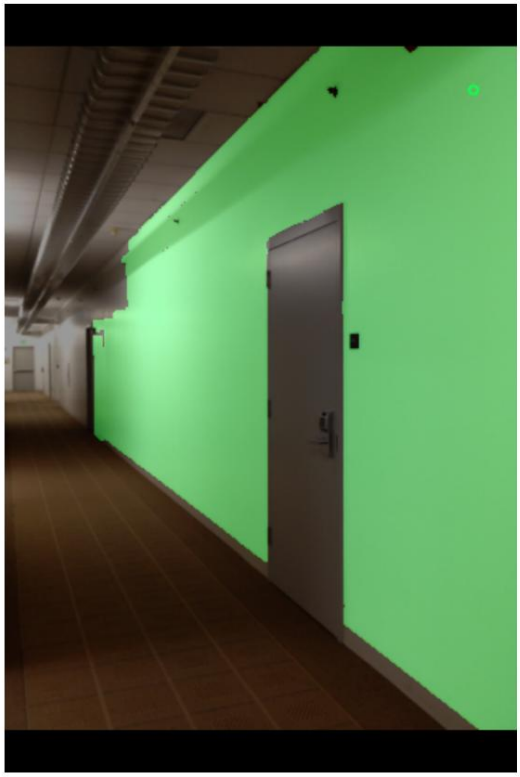
Tabel 2. Durasi setiap langkah pemrosesan.

1. Membuat citra RGB dari citra RGBA 2.	10 ms
Mengubah citra berwarna menjadi citra grayscale untuk deteksi tepi 3. Deteksi	10 ms
tepi Canny 4. Menyalin hasil Canny ke citra RGB untuk (7)	80 ms
	10 ms
5. Banjir mengisi	5 ms
6. Menyalin warna dinding target ke gambar topeng	5 ms
7. Menghapus tepi dari topeng 8. Erosi topeng untuk	40 ms
pelacakan 9. Pencampuran alfa	50 ms
	40 ms

Gambar 3 a,b,c,d.



Gambar 4.



Gambar 5.

ambang batas di tepi sehingga hanya satu bidang dinding yang akan disegmentasi pada satu waktu, dan kemudian memungkinkan pengguna untuk secara manual memilih beberapa titik benih yang sesuai dengan beberapa bidang dinding yang ingin diwarnai oleh pengguna. Ini dianggap terlalu rumit dari antarmuka pengguna untuk diterapkan pada prototipe ini, sehingga ambang batas yang lebih lemah diizinkan untuk dapat mewarnai di beberapa bidang dinding.

B. Pelacakan

Algoritme mampu melacak wilayah dinding dengan baik di sebagian besar keadaan; namun, titik benih kadang-kadang akan melompat dari satu permukaan dinding ke permukaan lainnya (misalnya dari dinding samping ke langit-langit) ketika titik benih terlalu dekat dengan perbatasan, dan ini dapat menyebabkan beberapa kedipan. Hal ini kemungkinan disebabkan oleh penyebab yang sama seperti yang dijelaskan pada bagian segmentasi, bahwa batas tepi dinding yang cukup lemah memungkinkan titik benih melompat ketika dua wilayah terdeteksi sebagai satu wilayah untuk sesaat. Ini harus diatasi dalam iterasi berikutnya dari aplikasi.

Cara kedua untuk meningkatkan pelacakan adalah dengan meningkatkan kecepatan bingkai, karena ini meningkatkan validitas asumsi bahwa dinding akan tetap berada di lokasi yang kira-kira sama dari bingkai ke bingkai. Hal ini karena pengurangan waktu antar bingkai akan mengurangi perpindahan dinding antar bingkai, sehingga sebuah titik di topeng yang terkikis lebih mungkin menjadi bagian dari dinding di bingkai berikutnya. Oleh karena itu, pelacakan akan ditingkatkan.

C. Transformasi Warna

Meskipun ini bukan fokus dari karya yang disajikan dalam makalah ini, metode alpha-blending yang digunakan untuk transformasi warna dinding tersegmentasi terbukti menyenangkan dan realistis, menggambarkan bayangan secara akurat. Gambar 6 menunjukkan contoh segmen dinding dengan pilihan warna yang berbeda.

V. KESIMPULAN

Dalam karya ini, kami mendemonstrasikan prototipe aplikasi seluler yang berhasil dikembangkan untuk iOS yang dapat menyegmentasikan, melacak, dan mewarnai dinding pada kecepatan bingkai video. Uji coba menunjukkan algoritme untuk dapat bekerja dengan baik di sebagian besar keadaan, dengan penurunan kekokohan pada pemandangan beberapa bentuk dinding yang tidak sederhana atau dinding yang berpotongan. Penyesuaian parameter dalam tahapan pipeline seperti tahap deteksi tepi Canny (untuk meningkatkan kekuatan batas dinding) dapat meningkatkan kinerja, dan kami berencana untuk mengatasi hal ini dalam pekerjaan aplikasi di masa mendatang. Antarmuka pengguna yang ditingkatkan yang memungkinkan pengguna untuk memilih beberapa wilayah dinding juga akan berguna setelah batas dinding yang lebih kuat ini membatasi area berwarna di wilayah dinding multi-bidang.

Kecepatan aplikasi dapat ditingkatkan lebih jauh dengan menerapkan lebih banyak alur algoritmik pada GPU sebagai pengganti CPU. Terakhir, kita dapat melihat metode transformasi warna yang memperkirakan kondisi pencahayaan untuk menyajikan pratinjau yang lebih akurat dari warna cat yang diinginkan.

REFERENSI

[1] Lee, James, R Haralick, dan Linda Shapiro. "Deteksi tepi morfologis." *Robotika dan Otomasi, IEEE Journal of* 3.2 (1987): 142-156.

[2] Magarey, Julian, dan Nick Kingsbury. "Estimasi gerak menggunakan transformasi wavelet bernilai kompleks." *Pemrosesan Sinyal, Transaksi IEEE pada* 46.4 (1998): 1069-1084.

[3] Boykov, Yuri, Olga Veksler, dan Ramin Zabih. "Peminimalan energi perkiraan cepat melalui pemotongan grafik." *Analisis Pola dan Kecerdasan Mesin, Transaksi IEEE pada* 23.11 (2001): 1222-1239.

[4] Takacs, Gabriel dkk. "Augmented reality di luar ruangan pada ponsel menggunakan organisasi fitur visual berbasis loxel." *Prosiding konferensi internasional ACM pertama tentang pencarian informasi Multimedia* 30 Oktober 2008: 427-434.

[5] Friedman, Nir, dan Stuart Russell. "Segmentasi gambar dalam urutan video: Pendekatan probabilistik." *Prosiding konferensi Ketigabelas tentang Ketidakpastian dalam kecerdasan buatan* 1 Agustus 1997: 175-181.

[6] Zeng, Bing, Renxiang Li, dan Ming L Liou. "Optimasi algoritma estimasi gerakan blok cepat." *Sirkuit dan Sistem untuk Teknologi Video, Transaksi IEEE pada* 7.6 (1997): 833-844.

[7] Takacs, Gabriel dkk. "Pelacakan dan pengenalan waktu nyata terpadu dengan fitur cepat rotasi-invarian." *Computer Vision and Pattern Recognition (CVPR), Konferensi IEEE 2010 pada* 13 Juni 2010: 934-941.

[8] Klein, Georg, dan David Murray. "Pelacakan dan pemetaan paralel pada ponsel kamera." *Mixed and Augmented Reality, 2009. ISMAR 2009. Simposium Internasional IEEE ke-8 pada* 19 Oktober 2009: 83-86.

[9] https://vimeo.com/4340549, kata sandi 'nana'

[10] http://www.opengl.org/wiki/FAQ

[11] https://github.com/BradLarson/GPUImage

Gambar 6. Dinding di bawah warna terapan yang berbeda.

LAMPIRAN A: KONTRIBUSI

Bagian ini menjelaskan kontribusi masing-masing individu dalam tim.

Serena Yeung

- Pengembang iOS - berfokus pada integrasi CPU/GPU dan penyesuaian warna
- Mengambil kode C++ pelacak RIFF dari TA dan men-debug/ mengkompilasinya untuk dijalankan di PC. Itu akhirnya dinilai tidak perlu untuk aplikasi, yang tidak memerlukan pengenalan fitur.
- Kerangka kerja GPUImage terintegrasi ke dalam proyek dan mekanisme komunikasi yang diterapkan antara CPU dan GPU
- Kerangka kerja OpenCV terintegrasi ke dalam proyek
- Menerapkan UI pemilih warna
- Penulisan laporan

Jeff Piersol

- Pengembang iOS - berfokus pada segmentasi waktu nyata dan pengembangan pelacakan
- Masalah kebocoran memori yang di-debug di iOS
- Modularisasi kode Objective C untuk keterbacaan dan kinerja
- Menerapkan algoritme pelacakan titik benih di iOS
- Penulisan laporan: pengoptimalan algoritme dan kinerja

David Liu

- Desain algoritme - berfokus pada perancangan algoritme yang berfungsi
- Menghubungi dan berinteraksi dengan TA dan Profesor Girod untuk kemungkinan ide algoritme untuk bereksperimen
- Meneliti makalah IEEE yang relevan
- Membuat video saluran di poster
- Membuat poster

LAMPIRAN B: ALGORITMA DIPERTIMBANGKAN

Algoritme berikut adalah beberapa algoritme tonggak utama:
yang dicoba tetapi tidak termasuk dalam pipa akhir proyek. Meskipun algoritma ini tidak digunakan, mereka membantu dalam pengembangan algoritma yang diimplementasikan.

A. Analisis Gambar Video Awal

- Histogram skala abu-abu
- Ambang batas Otsu Global
- Ambang batas adaptif dari titik benih

Algoritme ini memberikan wawasan tentang karakteristik bingkai video. Salah satu temuan utama adalah kamera melakukan penyesuaian kecerahan otomatis. Oleh karena itu, penting untuk tidak hanya mengandalkan nilai iluminansi gambar dari satu frame ke frame berikutnya. Hal ini terlihat melalui video histogram grayscale.

B. Metode Pelacakan berbasis fitur

Metode berbasis fitur dieksploitasi lebih lanjut untuk melihat apakah ada fitur yang dapat dihitung dalam kecepatan bingkai video pada resolusi 480 x 360 piksel. Detektor fitur cepat rotasi-invarian menjanjikan untuk mencapai kriteria tersebut [1]. Dengan demikian, kode C++ pelacakan RIFF eksperimental diperoleh dari penulis RIFF dan pengujian dilakukan untuk menguji kelayakan algoritma. Namun, itu tidak cukup cepat untuk iOS penerapan.

C. Metode Pelacakan Sederhana

Berbagai metode pelacakan berbasis fitur dicoba: pelacakan dengan centroid, pelacakan pusat lambung cembung, dan kuantitas geometris sederhana. Namun, segmen tersebut belum tentu cembung dan tidak menjamin bahwa seed point frame berikutnya berada di segmen yang sama.

D. Deteksi tepi:

- Laplacian dari deteksi tepi Gaussian
- Filter Sobel horizontal/vertikal
- Deteksi tepi yang cerdik

Di atas adalah algoritma deteksi tepi terkenal dari literatur yang diuji dengan berbagai parameter. Deteksi tepi Canny telah terbukti menjadi hasil terbaik dalam situasi ini.
Peningkatan lebih lanjut dilakukan melalui penyaringan hit-miss, kemudian dilasi untuk menutup celah.

- Pemfilteran hit-miss:
  - Hapus satu noise piksel
  - Hapus noise dua piksel (vertikal dan horizontal)
  - Hapus semua piksel yang ada di dalam kotak pembatas 3x3

Operasi ini menghilangkan kebisingan yang signifikan dan menutup beberapa celah tepi. Namun, mereka relatif mahal secara komputasi dan tidak termasuk dalam algoritma akhir.

E. Transformasi Hough

Karena ada celah yang hilang dalam algoritma deteksi tepi, garis Hough ditemukan mencoba untuk mengisi celah secara akurat. Namun, hasilnya tidak akurat dan terlalu mahal secara komputasi.

F. Algoritma Penerangan dan Penyesuaian Warna

Seluruh dinding diasumsikan memiliki reflektansi spektral yang sama. Selanjutnya, diasumsikan bahwa gambar menyala dengan sumber cahaya tunggal; spektrum iluminasi pada setiap piksel hanya dapat berbeda dengan skalar. Jadi, dengan menggunakan nilai RGB terukur, upaya dilakukan untuk mendapatkan estimasi akurat dari spektrum iluminan. Namun, implementasi estimasi tersebut tidak menghasilkan hasil yang berguna dalam MATLAB
simulasi. Salah satu masalah utama adalah bahwa kamera berlaku penyeimbangan warna otomatis pada nilai RGB mentah berdasarkan pencahayaan. Pekerjaan lebih lanjut dapat dilakukan di area ini untuk menyempurnakan hasil.