DETEKSI KELILING LUKA KRONIS MENGGUNAKAN *ACTIVE CONTOUR* (*SNAKE*) DAN *ACTIVE CONTOUR* YANG DITAMBAHKAN INTERPOLASI

Skripsi

Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer

Oleh:
Muhamad Rizki
3145160661

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA

2022

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul **"Deteksi Keliling Luka Kronis menggunakan *Active Contour* (*snake*) dan *Active Contour* yang ditambahkan *Preprocessing* Interpolasi"** yang disusun sebagai syarat untuk memperoleh gelar Sarjana komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Sumber informasi yang diperoleh dari penulis lain yang telah dipublikasikan yang disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

Jika dikemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian-bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang berlaku.

Bogor, 17 Agustus 2022

Muhamad Rizki

HALAMAN PERSEMBAHAN

Untuk Bapak, Ibu, dan Keluargaku

KATA PENGANTAR

Atas berkat rahmat Allah Yang Maha Kuasa, penulis bisa menyelesaikan skripsi yang berjudul **"Deteksi Keliling Luka Kronis menggunakan *Active Contour* (*snake*) dan *Active Contour* yang ditambahkan *Preprocessing* Interpolasi"**. Selain itu penulis juga berterima kasih kepada pihak-pihak pendukung dan mendoakan semoga kebaikan dan jasa-jasa yang sudah diberikan dibalas Allah Yang Maha Kuasa. Adapun pihak-pihak tersebut sebagai berikut:

1. Yth. Ibu Ir. Fariani Hermin Indiyah, M. T. selaku Koordinator Program Studi S-1 Ilmu Komputer Universitas Negeri Jakarta sekaligus dosen pembimbing akademik yang sudah membimbing penulis dalam hal akademik,
2. Yth. Bapak Med Irzal, M. Kom. selaku dosen pembimbing I yang banyak memberikan bantuan, masukan, dan saran baik secara konten maupun penulisan,
3. Yth. Bapak Muhammad Eka Suryana, M. Kom. selaku dosen pembimbing II yang banyak memberikan bantuan, masukan, dan saran baik secara konten maupun penulisan,
4. Yth. Ibu Ns. Ratna Aryani, M.Kep. beserta tim peneliti yang telah menyediakan dataset luka untuk untuk penulis,
5. Yth. Seluruh dosen program studi S-1 Ilmu Komputer yang sudah memberikan banyak ilmu kepada penulis selama perkuliahan,
6. Yth. Kedua orang tua penulis yang selama ini telah senantiasa sabar membimbing, memberikan semangat, mengingatkan, dan mendo'akan penulis,

7. Yth. Teman-teman Ilmu Komputer angkatan 2016 yang senantiasa menemani, memberikan semangat, dan motivasi dari semenjak awal dunia perkuliahan,

Merupakan kekurangan dari pribadi penulis jika masih ditemukan banyak kesalahan di dalam ini. Penulis sangat berterima kasih jika terdapat saran-saran membangun terkait skripsi penulis.

Depok, 31 Januari 2022

Muhamad Rizki

ABSTRAK

Muhamad Rizki. **Deteksi Keliling Luka Kronis menggunakan *Active Contour* (*snake*) dan *Active Contour* yang ditambahkan *Preprocessing* Interpolasi**. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2022. Di bawah bimbingan Med Irzal, M. Kom. dan Muhammad Eka Suryana, M. Kom

Luka kronis menjadi permasalahan bagi perawat luka dan instansi kesehatan terkait. Salah satu hal mendasar dalam penyembuhan luka kronis adalah melihat ukuran luka yang akan diamati dalam proses *assesment* luka yang saat ini masih dilakukan secara manual dan hal tersebut rentan dengan ketidakakuratan. Untuk mengatasi ketidakakuratan pengukuran manual, maka metode pengukuran keliling luka berbasis analisa citra (image), khususnya citra biomedis (*biomedical image*) dan citra medis (*medical image*) perlu dikembangkan. Skripsi ini bertujuan untuk mengimplementasi dan melihat hasil metode *active contour* (*snake*) dalam kasus deteksi keliling luka kronis. Implementasi yang dikembangkan menggunakan metode *snake* dengan *snake* yang ditambahkan interpolasi untuk deteksi keliling luka kategori luka hitam, kuning, dan merah. Hasil akhir menunjukkan bahwa data yang berhasil dideteksi menggunakan *snake* interpolasi (44 data dari 71 data) lebih banyak dibandingkan versi *integer* (12 data dari 71 data) dengan nilai akurasi rata-rata 77.18% untuk *snake* versi integer dan 86.1% untuk *snake* versi interpolasi

Kata kunci : Luka kronis, *active contour*, deteksi keliling, interpolasi.

ABSTRACT

Muhamad Rizki. **Detection of Chronic Wound Circumference using Active Contour(snake) and Active Contour with added Interpolation**. Thesis. Faculty of Mathematics and Natural Sciences, State University of Jakarta. 2022. Under the guidance of Med Irzal, M. Kom. and Muhammad Eka Suryana, M. Kom

Chronic wounds are a problem for wound nurses and related health agencies. One of the basic things in chronic wound healing is to see the size of the wound that will be observed in the wound assessment process, which is currently still done manually and is prone to inaccuracies. To overcome the inaccuracy of manual measurements, the method of measuring wound circumference based on image analysis, especially biomedical images and medical images, needs to be developed. This thesis aims to implement and see the results of the active contour (snake) method in the case of chronic wound circumference detection. The implementation developed uses the snake method with a snake added by interpolation to detect the circumference of the wound in the black, yellow, and red categories. The final result shows that the data detected using snake interpolation (44 data from 71 data) is more than the integer version (12 data from 71 data) with an average accuracy value of 77.18% for the integer version of the snake and 86.1% for the interpolated version of the snake.

key word : Chronic wound, active contour, circumference detection, interpolation

DAFTAR ISI

HALAMAN PERSEMBAHAN iii

KATA PENGANTAR iv

ABSTRAK vi

ABSTRACT vii

DAFTAR ISI x

DAFTAR GAMBAR xiii

DAFTAR TABEL xvi

I PENDAHULUAN 1

1.1 Latar Belakang Masalah 1

1.2 Rumusan Masalah 7

1.3 Pembatasan Masalah 7

1.4 Tujuan Penelitian 8

1.5 Manfaat Penelitian 8

II KAJIAN PUSTAKA 9

2.1 Populasi dan Sampel 9

2.2 Pengolahan Citra Digital (*Digital Image Processing*) 10

2.2.1 Citra *Grayscale* 10

2.2.2 *Gaussian Low Pass Filter* / *Gaussian Filter* 12

2.3 Gradien citra (*image gradient*) 13

2.4 *Active contour* 15

2.4.1 Representasi *Snake* 15
2.4.2 Energi internal 17
2.4.3 Energi eksternal 18
2.5 *Active contour evolution* 19
2.6 Interpolasi Bilinear 24

III METODOLOGI PENELITIAN **26**
3.1 Pengolahan data citra input 26
3.2 Deteksi keliling menggunakan *Active Contour* (*snake*) 29
3.2.1 Inisialisasi kurva awal *Active Contour* 29
3.3 *Ground truth* 30
3.4 Validasi 31

IV HASIL DAN PEMBAHASAN **32**
4.1 Pengolahan data citra input 32
4.2 Deteksi keliling luka menggunakan *active contour* (*snake*) 32
4.2.1 Inisialisasi kurva awal 33
4.2.2 Energi internal 34
4.2.3 Energi eksternal 35
4.2.4 Proses update intersi kurva 35
4.3 Eksperimen 40
4.4 Analisa hasil 41

V KESIMPULAN DAN SARAN **44**
5.1 Kesimpulan 44
5.2 Saran 44

DAFTAR PUSTAKA **49**

A *Source code* program **50**

1.1 main_integer.py 50

1.2 main_interpolation.py 53

1.3 integer_validation.py 56

1.4 interp_validation.py 57

B Tabel pengolahan data citra input **59**

C Tabel hasil deteksi keliling luka menggunakan *snake* versi *integer* **71**

D Tabel hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* **86**

DAFTAR GAMBAR

Gambar 2.1 Representasi Citra Digital 10

Gambar 2.2 Contoh citra *grayscale* 11

Gambar 2.3 Proses konversi citra RGB menjadi *grayscale* 12

Gambar 2.4 (a) citra *grayscale* ; (b) $|G_x * I(x, y)|$, gradien arah x menggunakan kernel Sobel-x ; (c) $|G_y * I(x, y)|$, gradien arah x menggunakan kernel Sobel-y ; (d) gradien citra, $|G_x * I(x, y)| + |G_y * I(x, y)|$ (Gonzalez and Woods, 2002) . 14

Gambar 2.5 (a) citra objek lingkaran & *initial snake*, (b) evolusi kurva *snake*, (c) bentuk akhir dari *snake* setelah iterasi selesai (Acton and Ray, 2007) 15

Gambar 2.6 Kurva ingkaran 16

Gambar 2.7 *source code* inisialisasi kurva lingkaran 16

Gambar 2.8 Aproksimasi turunan dengan *finite differences* (Ivins and Porrill, 1995) 20

Gambar 2.9 interpolasi bilinear 25

Gambar 3.1 Diagram alir penelitian 26

Gambar 3.2 (a) Data citra format .xcf, (b) *layer* citra (luka), (c) *layer* region, (d) *path* 27

Gambar 3.3 Citra luka yang telah dicek menggunakan fitur *eclipse select* dan ditambahkan *border* 28

Gambar 3.4 Citra luka dan region luka 28

Gambar 3.5 Diagram alir *snake* 29

Gambar 3.6 (A) citra luka, (b) citra luka (*grayscale*), (c) inisialisasi kurva awal dengan cr = 120 , cc = 265, r = 85 30

Gambar 3.7 Komparasi Citra luka, region luka, dan *groundtruth* 30

Gambar 4.1 *source code* konversi citra RGB ke *grayscale* 32
Gambar 4.2 *source code* inisialisasi kurva lingkaran versi *integer* 33
Gambar 4.3 *source code* inisialisasi kurva lingkaran versi interpolasi . . . 33
Gambar 4.4 *source code* untuk menghasilkan energi internal 34
Gambar 4.5 *source code* energi eksternal untuk *snake* versi *integer* 35
Gambar 4.6 *source code* energi eksternal untuk *snake* versi interpolasi . . 35
Gambar 4.7 *source code* proses update iterasi kurva versi *integer* 36
Gambar 4.8 *source code* proses update iterasi kurva versi interpolasi . . . 37
Gambar 4.9 Bentuk kurva *snake* terdiri dari pasangan koordinat dengan nilai *float* 38
Gambar 4.10 komparasi gradien citra arah (*direction*) y (`gy`). (a) citra asli (b) citra hasil interpolasi 39
Gambar 4.11 (a) Berhasil (b) Gagal 40
Gambar 4.12 (a) Hasil deteksi (b) Hasil deteksi *overlay* dengan *ground truth* (c) Area kurva akhir (d) Area *ground truth* (e) Akurasi . 41
Gambar 4.13 (a) Hasil *snake* versi integer dengan pembulatan kurva di luar dan di dalam iterasi (b) Hasil *snake* versi integer dengan pembulatan kurva hanya di dalam iterasi 43
Gambar 4.14 *source code* proses update iterasi kurva versi *integer* dengan pembulatan kurva hanya di dalam iterasi 43

Gambar 3.1 parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka hitam . 84
Gambar 3.2 parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka kuning 84

Gambar 3.3 parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka merah . 85

Gambar 4.1 parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka hitam . 99

Gambar 4.2 parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka kuning 100

Gambar 4.3 parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka merah . 101

DAFTAR TABEL

Tabel 2.1 Visualisasi hasil pengolahan data citra input luka hitam . . . 59

Tabel 2.2 Visualisasi hasil pengolahan data citra input luka hitam - lanjutan . . . 60

Tabel 2.3 Visualisasi hasil pengolahan data citra input luka hitam - lanjutan . . . 61

Tabel 2.4 Visualisasi hasil pengolahan data citra input luka hitam - lanjutan . . . 62

Tabel 2.5 Visualisasi hasil pengolahan data citra input luka kuning . . . 63

Tabel 2.6 Visualisasi hasil pengolahan data citra input luka kuning - lanjutan . . . 64

Tabel 2.7 Visualisasi hasil pengolahan data citra input luka kuning - lanjutan . . . 65

Tabel 2.8 Visualisasi hasil pengolahan data citra input luka merah . . . 66

Tabel 2.9 Visualisasi hasil pengolahan data citra input luka merah - lanjutan . . . 67

Tabel 2.10 Visualisasi hasil pengolahan data citra input luka merah - lanjutan . . . 68

Tabel 2.11 Visualisasi hasil pengolahan data citra input luka merah - lanjutan . . . 69

Tabel 2.12 Visualisasi hasil pengolahan data citra input luka merah - lanjutan . . . 70

Tabel 3.1 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* . . . 72

Tabel 3.2 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 73
Tabel 3.3 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 74
Tabel 3.4 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 75
Tabel 3.5 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 76
Tabel 3.6 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 77
Tabel 3.7 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 78
Tabel 3.8 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 79
Tabel 3.9 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 80
Tabel 3.10 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 81
Tabel 3.11 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 82
Tabel 3.12 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan 83
Tabel 4.1 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* 87
Tabel 4.2 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* 88

Tabel 4.3 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 89
Tabel 4.4 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 90
Tabel 4.5 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 91
Tabel 4.6 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 92
Tabel 4.7 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 93
Tabel 4.8 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 94
Tabel 4.9 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 95
Tabel 4.10 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 96
Tabel 4.11 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 97
Tabel 4.12 Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan 98

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Salah satu organ yang sangat penting pada tubuh manusia adalah kulit. Ketika kulit terluka, diperlukan perawatan luka yang baik agar tidak terjadi infeksi. Luka adalah keadaan di mana fungsi anatomis kulit normal mengalami kerusakan akibat proses patologis yang berasal dari internal maupun eksternal dan mengenai organ tertentu. Proses penyembuhan luka terjadi melalui beberapa fase, yaitu: hemostasis (beberapa jam pasca-terjadinya luka), inflamasi (1 – 3 hari), proliferasi (4 – 21 hari), dan *remodelling* (21 hari – 1 tahun). Fase-fase penyembuhan luka terjadi secara bertahap, namun dapat terjadi secara bersamaan (*overlap*) (Simon, 2018).

Luka kronis adalah kondisi di mana luka mengalami proses penyembuhan yang tidak normal dengan durasi fase-fase yang sesuai (Landén et al., 2016), kondisi ini dapat berkaitan dengan berbagai faktor yang memperlambat penyembuhan luka seperti adanya penyakit kronis, insufisiensi vaskuler, diabetes, gangguan nutrisi, penuaan, dan berbagai faktor lokal pada luka (tekanan, infeksi, dan edema). Secara umum, luka kronis dapat terjadi akibat ulkus vena, ulkus arteri, ulkus dekubitus, dan ulkus diabetik (Zhao et al., 2016).

Setiap tahunnya prevalensi luka secara umum mengalami peningkatan. 1.4 juta orang dewasa dirawat karena luka kekerasan di tahun 2000 sampai 2010, 1,6% di antaranya dirawat di Unit Gawat Darurat (UGD) di Amerika Serikat (Monuteaux et al., 2017). Berdasarkan hasil Riset kesehatan dasar (Riskesdas) tahun 2018, prevalensi luka di Indonesia selalu mengalami peningkatan dari tahun 2007 sebanyak 7,5%, tahun 2013 sebanyak 8,2%, dan di tahun 2018 sebanyak 9.2%

(Kesehatan, 2018).

Luka kronis bisa jadi merupakan komplikasi pada penderita Diabetes Melitus (DM). Sebanyak 15% dari seluruh populasi penderita DM memiliki komplikasi berupa luka diabetes (Fard et al., 2007). Prevalensi Diabetes Melitus di Indonesia kategori penduduk umur 15 tahun keatas pada tahun 2018 adalah 2% (Kesehatan, 2018). Pada tahun 2030 diprediksi meningkat menjadi 21,3 juta orang. Indonesia menempati peringkat keempat jumlah penderita DM terbanyak di dunia (Wild et al., 2004).

Berdasarkan hasil prevalensi, luka kronis menjadi permasalahan bagi perawat luka dan instansi kesehatan terkait. Selain itu diperlukan penanganan khusus dalam proses pemulihan luka kronis. Permasalahan luka kronis menghadirkan kesulitan berat bagi yang menderita kondisi ini dan beban keuangan untuk industri kesehatan. Di sisi klien dapat berdampak pada penurunan kualitas hidup, ketidakmampuan untuk melakukan fungsi tubuh secara optimal, serta tingginya kebutuhan finansial. Bahkan dalam beberapa kasus dapat menyebabkan amputasi dan kematian. Bagi instansi kesehatan terkait akan memberikan dampak pada tingginya pembayaran asuransi kesehatan dikarenakan frekuensi perawatan luka yang dilakukan paling tidak 2,4 kali per minggu di mana menghabiskan 66% waktu perawat luka (HSE, 2007). Di Amerika Serikat , luka kronis setiap tahunnya menelan biaya $20 miliar dan memengaruhi 5,7 juta orang (Brown et al., 2018). Berdasarkan data dari alodokter.com, biaya perawatan luka di Indonesia berkisar mulai dari antara Rp61.500,00–Rp267.000,00 belum termasuk biaya balutan.

Salah satu tanggung jawab perawat luka profesional adalah melakukan pengkajian pada luka, di mana hasil pengkajian tersebut bermanfaat untuk menentukan pemberian balutan luka yang tepat, memonitor perbaikan luka dan mencegah komplikasi sehingga perawatan akan menjadi *cost effective* (Benbow,

2016).

Salah satu hal yang mendasar dalam proses penyembuhan luka kronis adalah melihat ukuran luka, dan umumnya hal ini adalah kriteria pertama yang harus dipertimbangkan dalam proses *assessment* luka yang mana hal ini memiliki peran penting di antaranya memantau laju dan kemajuan penyembuhan, mengevaluasi efektivitas perawatan, dan mengidentifikasi luka. Selain itu perubahan ukuran luka memungkinkan kita dalam mengamati tingkat penutupan, waktu penutupan, pelebaran, dan wawasan lain yang merupakan indikator untuk memprediksi status penyembuhan (Carrión et al., 2022). Masih digunakannya metode konvensional seperti mengukur menggunakan penggaris memiliki tingkat akurasi dan reliabilitas rendah sehingga perawat terkesan lambat dalam memberikan perawatan dibandingkan dengan profesi kesehatan yang lain seperti dokter. Sebuah studi menyatakan standar teknik pengukuran perawatan yang melibatkan penggunaan perkiraan penggaris dan mata telanjang, memiliki tingkat kesalahan 44% (Budman et al., 2015). Untuk mengatasi ketidakakuratan pengukuran manual, maka metode pengukuran keliling luka berbasis analisa citra (*image*), khususnya citra biomedis (*biomedical image*) dan citra medis (*medical image*) perlu dikembangkan.

Saat ini penelitian *state of the art* telah dilakukan untuk analisis luka. Banyak studi yang menjelaskan evaluasi luka menggunakan citra luka untuk mengetahui status luka. Citra medis dapat mengirimkan informasi lebih banyak untuk para ahli kesehatan daripada deskripsi subjektif yang cenderung menimbulkan kesalahan interpretasi. Lebih jauh lagi, gambar citra luka dapat digunakan untuk mentransmisi informasi tentang status penyembuhan untuk konsultasi medis di lokasi pedalaman. Pada sebuah percobaan tahun 2013, ditemukan nilai tinggi yang dihubungkan ke galeri foto luka di aplikasi mobile dan pelacakan luka melalui progresi grafis. Sehingga, untuk meningkatkan fitur citra diperlukan pengembangan algoritma

analisis citra untuk penentuan ukuran dan warna dari foto luka yang diambil dari kamera telepon pintar atau kamera tablet (Poon and Friesen, 2015a).

Ketika sebuah foto diambil dengan pose yang sama oleh kamera yang berbeda, warna yang tersimpan mungkin berbeda, ini merupakan salah satu penelitian paling awal untuk analisis luka. Salah satu solusi untuk masalah tersebut adalah menggunakan format *device independent* sRGB. Semua vendor kamera setidaknya menawarkan mode ini tetapi default ke format RGB mereka sendiri. Menggunakan *chart* referensi warna ketika gambar citra luka diambil, Poucke et. al., berhasil melakukan mekanisme kalibrasi antara perangkat *device dependent* RGB ke sRGB dengan mentransformasi citra terlebih dahulu menjadi ruang warna CIE *colorimetric*, kemudian ke sRGB melalui serangkaian masalah optimisasi (Van Poucke et al., 2010).

Upaya untuk menentukan batas luka oleh kamera telah dilaporkan dalam beberapa penelitian. Wang et. al. mengembangkan kotak pengambilan gambar dengan kamera *smartphone* & dua cermin, untuk menangkap gambar ulkus kaki dasar. Batas selanjutnya disempurnakan dengan algoritma *mean-shift* kemudian disetel lagi dengan *Region Adjacency Graph*. Setelah batas diperoleh, K-means dijalankan untuk mengukur rasio warna RYB. Metode ini dievaluasi pada 34 pasien yang berbeda di klinik Worcester. Kemudian penelitian Wang, masih berkonsentrasi pada penentuan batas luka tetapi dengan SVM (Wang et al., 2016). Pelabelan manual kumpulan data 100 luka yang menghasilkan 10.000 wilayah dilakukan oleh tim dokter (tiga ekspert) di sekolah kedokteran UMASS. Metode kerjanya sebagai berikut, pertama citra disegmentasi menjadi superpiksel dengan SLIC. Kemudian deskriptor warna & tekstur diekstraksi untuk persiapan setiap tahap SVM. Pada tahap pertama, warna & Tas kata diekstraksi dengan DSIFT. Selama tahap kedua, warna & tekstur *wavelet* diekstraksi. Tahap SVM pertama menjalankan k-binary

SVM *classifier* dilatih pada set citra yang berbeda. Pada tahap kedua, set kesalahan klasifikasi dilatih lagi dengan SVM biner. Setelah selesai, hasilnya sekali lagi disempurnakan dengan *Conditional Random Field*. Meskipun memberikan hasil yang memuaskan, semua percobaan diuji pada ulkus kaki (*close wound*) (Wang et al., 2014).

Friesen dari University of Manitoba memimpin tim peneliti untuk melakukan serangkaian penelitian dalam analisis luka. White et. al., sebagai tim peneliti pertama yang berkonsentrasi untuk mengukur ukuran luka ulkus tekan dilakukan dalam tiga langkah. Pertama, mengukur jarak dari kamera ke luka dengan referensi fokus. Kedua, kalibrasi pose kamera dari penggabungan data sensor (akselerometer, magnetometer & giroskop). Ketiga, Jepit & perbesar untuk mengukur ukuran luka dari referensi yang sebelumnya diketahui (White et al., 2014). Sayangnya, penyimpangan (*drift*) dari ukuran sebenarnya di atas cukup besar karena setiap langkah meningkatkan *drift*. Poon et. al., Yang melanjutkan penelitian mempertahankan fokus yang sama, kecuali jarak luka. Metode batas luka telah diubah menjadi *Grabcut*, maka setiap citra yang diambil diproyeksikan ke bidang 2d untuk menstabilkan sudut. Akhirnya warna tersegmentasi menjadi warna RYB (Poon and Friesen, 2015b). Salah satu kelemahan dari pendekatan yang diambil, deteksi batas dengan *Grabcut* hanya dapat dijalankan secara semi-otomatis, karena membutuhkan penyesuaian parameter.

Setelah batas luka telah ditentukan dengan benar, dapat digunakan untuk memproduksi *gel bioprinting* untuk penutupan luka. Gholami, *et. al.*, mengevaluasi tujuh algoritma untuk memenuhi tujuan ini (Gholami et al., 2017). Tiga algoritma yang dibandingkan adalah dari berbasis tepi, tiga lainnya berdasarkan pertumbuhan wilayah, dan satu lainnya berdasarkan tekstur. Algoritma *Livewire* yang didasarkan pada tepi adalah yang terbaik di antaranya. Berdasarkan kajian teori di atas, *grabcut*

digunakan sebagai segmentasi wilayah luka dan warna citra luka dikonversi menjadi *Commission internationale de l'éclairage* (CIE) untuk membuat mekanisme kalibrasi.

Salah satu metode yang banyak digunakan dalam aplikasi pemrosesan citra medis dan biomedis adalah metode kontur aktif (*active contour*) atau yang lebih dikenal dengan sebutan *snake* yang diperkenalkan oleh M. Kass et. al. pada tahun 1988 (Kass et al., 1988). Sebuah *active contour* (*snake*) adalah kurva yang meminimalkan fungsi energi untuk kondisi tertentu. Fungsi energi ini biasanya terdiri dari dua istilah: energi internal, yang membatasi kelancaran (*smoothness*) dan kekencangan (*tautness*) kontur, dan energi eksternal, yang menarik kontur elastis ke fitur-fitur menarik (Acton and Ray, 2007). Penelitian ini berfokus pada implementasi metode *snake* dalam mendeteksi keliling luka kronis. Ada kesulitan pada metode *snake* tradisional. Pertama kontur awal yang harus dekat dengan target, dengan kata lain jangkauan tangkap (*capture range*) *snake* terbatas. Masalah kedua ialah *snake* tradisional tidak dapat mengerakkan kontur ke dalam cekungan batas (*concave boundary*) (Guo et al., 2013) (Xu and Prince, 1998).

Abdullah, et. al. dalam penelitian tentang segmentasi iris berhasil mengatasi kelemahan dari metode *snake* dengan metode yang mereka usulkan, yaitu menambahkan *pressure force* pada *snake*. Arah pergerakan *snake* disesuaikan dengan kelopak mata sehingga menghasilkan hasil yang akurat dan efisien (Abdullah et al., 2016).

Penulis tertarik menerapkan metode *snake* pada penelitian ini dengan alasan metode ini adalah metode yang umum digunakan dalam aplikasi pemrosesan citra medis dan biomedis. Selain itu, *snake* cocok untuk mendeteksi objek dengan bentuk bebas *(free-form object)* seperti halnya dengan citra luka kronis. Dikutip dari jurnal *Medical and Biological Image Analysis* tahun 2018, *snake* telah banyak dan baik

digunakan dalam berbagai aplikasi seperti segmentasi CT-Scan otak, segmentasi untuk deteksi kanker payudara, deteksi lesi (bintik) pada kulit dan lain-lain (Hemalatha et al., 2018). Secara ideal, penulis ingin juga menerapkan metode yang dikembangkan oleh (Abdullah et al., 2016), namun hal tersebut belum dapat dilakukan karena harus memahami lebih terlebih dahulu tentang *snake*, maka dari itu untuk penelitian ini dibatasi menggunakan metode *active contour* saja. Penelitian ini akan berfokus pada pengembangan metode *snake* dalam mendeteksi keliling luka kronis.

Dataset citra yang penulis gunakan untuk penelitian ini didapat dari penelitian luka Ns. Ratna Aryani, M.Kep, tahun 2018 (Ratna Aryani, 2018) yang tersedia di *repository* `https://github.com/mekas/InjuryDetection`. Penelitian ini dilakukan sampai mendapatkan hasil berupa nilai akurasi yang didapat dari selisih area kurva akhir *snake* terhadap luas area *ground truth* (nilai sebenarnya).

1.2 Rumusan Masalah

Berdasarkan Latar belakang yang telah dikemukakan di atas. Fokus permasalahan pada penelitian ini adalah "Bagaimana cara mendeteksi keliling luka kronis menggunakan metode *Active contour* (*Snake*) dan *active contour* yang ditambahkan interpolasi ?".

1.3 Pembatasan Masalah

1. Pendeteksian keliling luka kronis menggunakan *snake* dan *snake* yang ditambahkan interpolasi menggunakan data citra luka yang didapat dari penelitian luka Ns. Ratna Aryani, M.Kep, tahun 2018 (Ratna Aryani, 2018).
2. Penelitian dilakukan sampai mendapatkan hasil, yaitu nilai akurasi dari selisih

area kurva *snake* terhadap area *ground truth*

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah untuk mengetahui hasil dari metode *snake* dan *snake* yang ditambahkan interpolasi dalam mendeteksi keliling luka kronis.

1.5 Manfaat Penelitian

1. Bagi peneliti

   Penelitian ini merupakan media penerapan ilmu pengetahuan, khususnya dalam pengembangan metode *Active contour* pada pengkajian luka kronis.

2. Instansi terkait

   Metode yang diajukan diharapkan dapat membuka peluang untuk diajukan ke instansi kesehatan terkait dalam proses pengkajian luka kronis.

3. Bagi ilmu pengetahuan

   - Mahasiswa

     Diharapkan penelitian ini dapat digunakan sebagai penunjang referensi, khususnya pustaka tentang deteksi keliling menggunakan *Active contour*.

   - Bagi peneliti selanjutnya

     Diharapkan Penelitian ini dapat digunakan sebgai dasar atau kajian awal bagi peneliti lain yang ingin meneliti permasalahan yang sama.

BAB II

KAJIAN PUSTAKA

2.1 Populasi dan Sampel

Tujuan dari sebuah riset adalah untuk memperoleh informasi dari populasi. Populasi merupakan seluruh kumpulan elemen yang dapat digunakan untuk membuat kesimpulan tertentu sedangkan sampel merupakan kelompok yang dipilih dari populasi untuk digunakan dalam riset. Sampai saat ini belum ada kesepakatan atau ketentuan secara ideal dalam menentukan berapa banyak sampel dalam penelitian. Sampel yang baik adalah sampel yang mencerminkan populasinya (Amirullah, 2015). Ukuran sampel harus diperhatikan dalam melakukan penelitian. Gay & Diehl berpendapat bahwa ukuran sampel harus sebesar-besarnya, semakin besar ukuran sampel maka akan semakin representatif, ukuran sampel yang dapat diterima bergantung pada jenis penelitiannya sebagai berikut (Gay and Diehl, 1992):

1. Penelitian deskriptif sampel minimumnya 10% dari populasi.
2. Penelitian yang bersifat korelasional sampel minimunnya 30 subyek
3. penelitian kausal-perbandingan, sampelnya sebanyak 30 subyek per grup
4. penelitian eksperimental, sampel minimunnya adalah 15 subyek per grup.

Fraenkel & Wallen menyarankan besar minimum untuk ukuran sampel sebagai berikut (Fraenkel et al., 2012):

1. Penelitian deskriptif sebanyak 100 sampel.
2. Penelitian yang bersifat korelasional sebanyak 50.
3. penelitian kausal-perbandingan sebanyak 30 per grup.

4. penelitian eksperimental sebanyak adalah 30 atau 15.

2.2 Pengolahan Citra Digital (*Digital Image Processing*)

Pengolahan citra digital adalah bidang ilmu yang mengacu pada pada pengolahan citra digital dengan menggunakan komputer digital. Citra digital terdiri dari sejumlah elemen yang masing-masing memiliki lokasi dan nilai tertentu. Elemen-elemen ini sering disebut dengan istilah piksel (Gonzalez and Woods, 2002). Citra dapat didefinisikan sebagai fungsi intensitas dua dimensi $I(x, y)$ yang direpresentasikan sebagai matriks berukuran m x n sebagai berikut:

$$I(x,y) = \begin{bmatrix} I(0,0) & I(0,1) & \cdots & I(0,n-1) \\ I(1,0) & I(1,1) & \cdots & I(1,n-1) \\ \vdots & \vdots & & \\ I(m-1,0) & I(m-1,1) & \dots & I(m-1,n-1) \end{bmatrix}$$

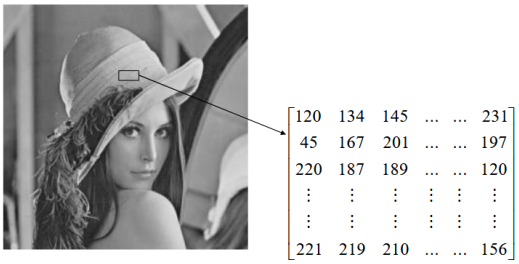
Gambar 2.1: Representasi Citra Digital

m x n menyatakan resolusi citra, dan setiap elemen dari matriks menyatakan sebuah piksel (*picture element*). Nilai I pada pasangan koordinat (x, y) disebut intensitas (*intensity*). Dalam operasi pengolahan citra, sebagian besar operasi dilakukan dalam citra *grayscale* (Tyagi, 2018).

2.2.1 Citra *Grayscale*

Citra *grayscale* adalah citra yang hanya memiliki satu kanal (*channel*) pada setiap pikselnya yang mewakili intensitas. Intensitas piksel berada dalam kisaran [0, 255] yang mana hal ini menunjukkan tingkat terangnya atau tingkat cahaya dari suatu pixel. Warna pada citra *grayscale* merupakan warna abu dengan tingkatan dari hitam

hingga sampai putih (tingkat keabuan). Kisaran intensitas piksel bernilai 0 artinya hitam dan 255 adalah putih (untuk *256-graylevel*).

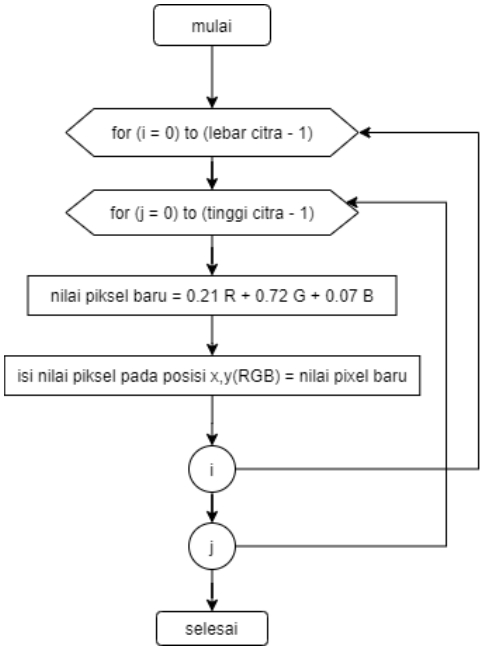


Gambar 2.2: Contoh citra *grayscale*

Setiap piksel dalam citra umumnya dijelaksan oleh kombinasi tiga nilai intensitas R (*red*), G (*green*),dan B (*blue*). Salah satu cara memetakan nilai tersebut ke satu nilai *grayscale* adalah dengan menggunakan metode *luminosity* (Kumar et al., 2016).

$$L(x, y) = 0.21R(x, y) + 0.72G(x, y) + 0.07B(x, y). \tag{2.1}$$

$L(x, y)$ melambangkan nilai intensitas *grayscale* pada pasangan koordinat (x, y). R, G, dan B masing-masing adalah nilai intensites citra *channel* R (*red*), G (*green*),dan B (*blue*) pada pasangan koordinat (x, y)



Gambar 2.3: Proses konversi citra RGB menjadi *grayscale*

2.2.2 *Gaussian Low Pass Filter / Gaussian Filter*

Peningkatan kualitas citra (*image enhancement*) adalah proses mengedit citra untuk membuatnya 'lebih baik' untuk aplikasi tertentu (Tyagi, 2018). Hal ini melibatkan proses menghaluskan atau mempertajam konten citra. Salah satu metode dalam proses menghaluskan citra adalah *filtering*, salah satunya adalah dengan menggunakan metode *Gaussian filter*. Proses ini adalah proses memblur citra menggunakan fungsi Gaussian dengan tujuan mengurangi *noise* citra dan mengurangi detail tertentu. Kernel gaussian *Gaussian blur* $G(x, y)$ dideskripsikan sebagai berikut:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

σ dan e menujukkan standar deviasi dan konstanta logaritma natural ($e \approx 2,718281828459$), x dan y adalah posisi koordinat Kernel gaussian.

2.3 Gradien citra (*image gradient*)

Tepi (*edge*) dalam ruang lingkup pengolahan citra digital mencirikan batas dari suatu objek dalam citra digital. Deteksi tepi (*edge detection*) merupakan metode identifikasi objek dalam citra berbasis tepi. Salah satu metode deteksi tepi yang umum digunakan adalah metode gradien citra (*image gradient*). Dalam konsep matematis, gradien dikenal sebagai turunan pertama (*first-order derivatives*). Gradien dari citra I dilambangkan dengan notasi ∇I dan di definisikan sebagai berikut:

$$\nabla I = \begin{bmatrix} \nabla I(x,y)_x \\ \nabla I(x,y)_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix} \tag{2.3}$$

di mana $\nabla I(x,y)_x$ dan $\nabla I(x,y)_y$ masing-masing adalah persamaan gradien citra I arah (*direction*) x dan y yang dapat dihitung menggunakan dua persamaan berikut:

$$\nabla I(x,y)_x = I(x+1,y) - I(x,y) \tag{2.4}$$

$$\nabla I(x,y)_y = I(x,y+1) - I(x,y) \tag{2.5}$$

untuk besarnya (*magnitude*) gradien I dapat dihitung menggunakan persamaan berikut:

$$M(x,y) = ||\nabla I(x,y)|| = \sqrt{(\nabla I(x,y)_x)^2 + (\nabla I(x,y)_y)^2} \tag{2.6}$$

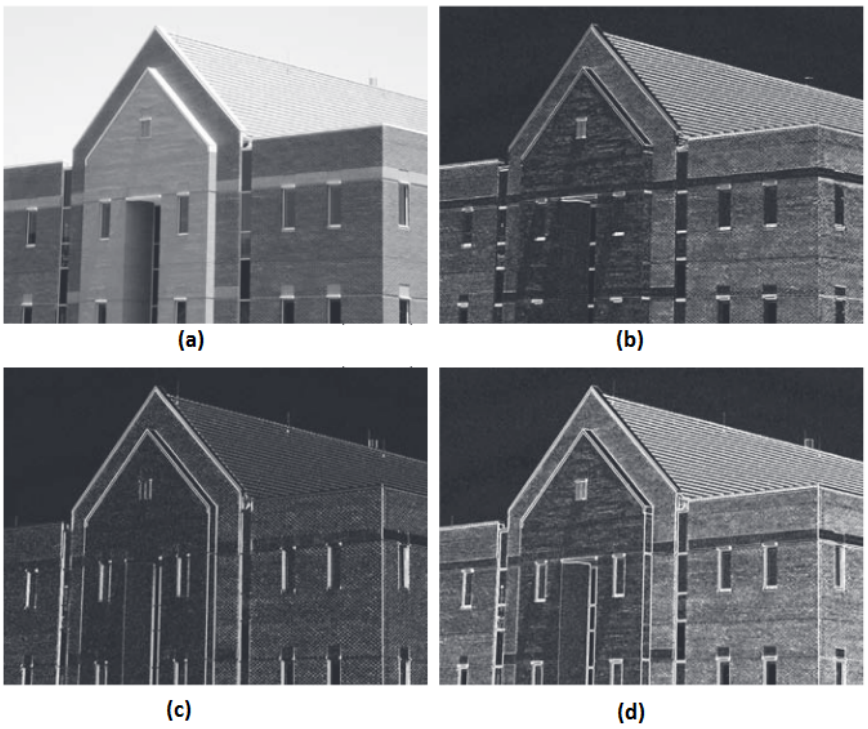
Selain menggunakan persamaan (2.4), (2.5), dan (2.6), perhitungan gradien citra dapat menggunakan operator gradien yang nantinya akan dikonvolusikan dengan citra. Salah satu operator gradien umum yang sering digunakan adalah operator Sobel/kernel Sobel. Konvolusi dilambangkan dengan $*$. Operator Sobel

arah x dan y, serta besar gradien Sobel masing-masing ditunjukkan oleh persamaan berikut:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.8}$$

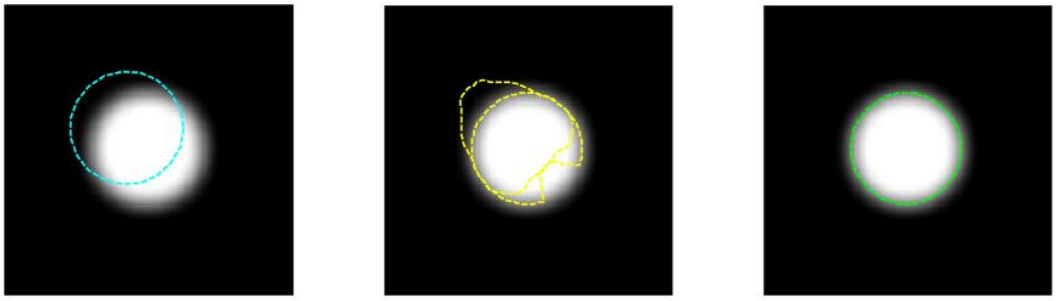
$$GS = |G_x * I(x,y)| + |G_y * I(x,y)| \tag{2.9}$$



Gambar 2.4: (a) citra *grayscale* ; (b) $|G_x * I(x,y)|$, gradien arah x menggunakan kernel Sobel-x ; (c) $|G_y * I(x,y)|$, gradien arah x menggunakan kernel Sobel-y ; (d) gradien citra, $|G_x * I(x,y)| + |G_y * I(x,y)|$ (Gonzalez and Woods, 2002)

2.4 *Active contour*

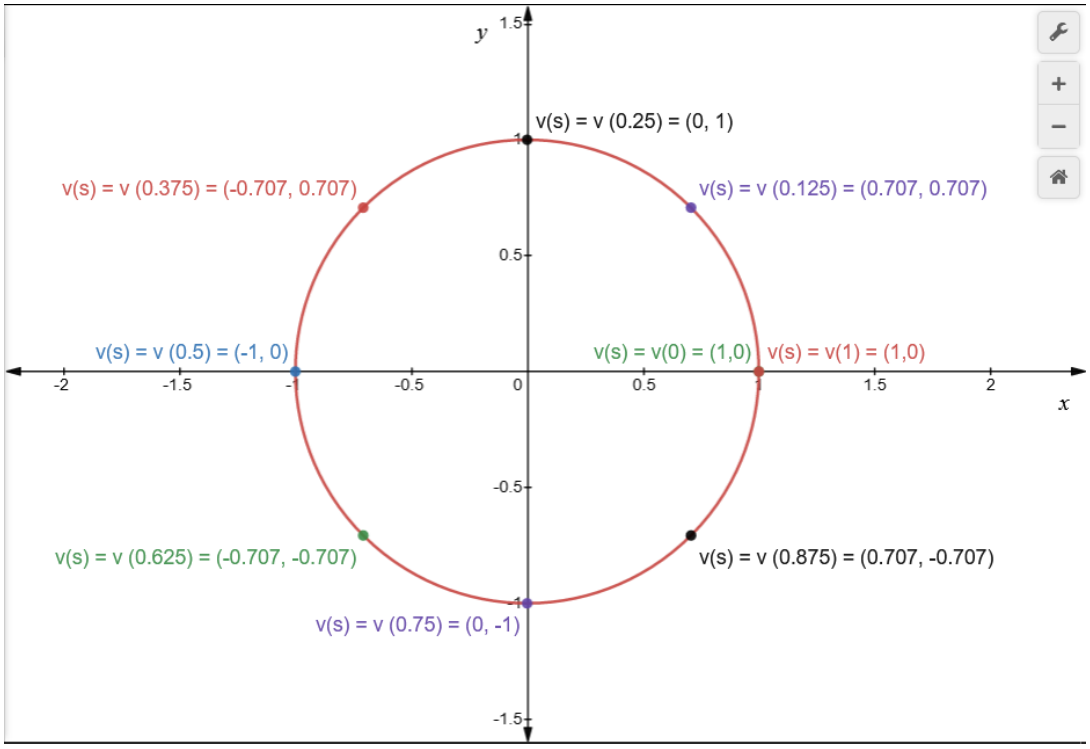
Active contour yang juga dikenal dengan sebutan *snakes* adalah kurva yang di definisikan dalam citra (*image*), kurva ini dapat bergerak menuju batas objek atau *feature* dari sebuah citra yang dipengaruhi oleh dua fungsional energi yang disebut energi internal dan energi eksternal.



Gambar 2.5: (a) citra objek lingkaran & *initial snake*, (b) evolusi kurva *snake*, (c) bentuk akhir dari *snake* setelah iterasi selesai (Acton and Ray, 2007)

2.4.1 Representasi *Snake*

Snake di representasikan sebagai kurva parametrik tertutup $\mathbf{v}(s) = (x(s), y(s))$, di mana s adalah panjang kurva dengan rentang tertentu, $x(s)$ dan $y(s)$ masing-masing merupakan elemen dari kurva $\mathbf{v}$ pada saat s. Sebagai contoh diberikan sebuah kurva sebagai berikut:



Gambar 2.6: Kurva ingkaran

Kurva di atas adalah kurva lingkaran $\mathbf{v}(s) = [cos(2\pi s), sin(2\pi s)]$, $s \in [0, 1]$. Perlu diperhatikan bahwa titik awal $\mathbf{v}(0)$ dan titik akhir $\mathbf{v}(1)$ di definisikan sebagai titik yang sama.

Algoritma pembuatan lingkaran menggunakan bentuk parametrik (*parametric form*) lingkaran adalah sebagai berikut:

```
repeat until theta >= 360;
{
        x = h + r*cos(theta)
        y = k + r*sin(theta)
        draw a line to x,y
        add step to theta
}
```

Gambar 2.7: *source code* inisialisasi kurva lingkaran

yang dilakukan algoritma di atas adalah menghasilkan koordinat `x, y` dari sebuah titik pada lingkaran yang diberi sudut (`theta`). Dimulai dari `theta = 0`

kemudian *looping* sampai `theta = 360` atau `theta = 2`π. `h` dan `k` adalah koordinat dari titik tengah lingkaran dan `r` adalah jari-jari lingkaran (Page, 2011)

Fungsional energi *snake* di definisikan sebagai berikut (Abdullah et al., 2016):

$$E_{snake} = \int_0^1 E_{int}(\mathbf{v}(s))ds + \int_0^1 E_{ext}(\mathbf{v}(s))ds. \tag{2.10}$$

2.4.2 Energi internal

Energi internal digunakan untuk mengontrol perubahan bentuk (deformabilitas) dari *snake*, yang ditulis sebagai berikut (Abdullah et al., 2016):

$$E_{int}(\mathbf{v}(s)) = \frac{1}{2}\left(\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2\right) \tag{2.11}$$

s pada $\mathbf{v}_s$ melambangkan turunan pertama, ss pada $\mathbf{v}_{ss}$ melambangkan turunan kedua, dan seterusnya. Fungsi energi internal ini terdiri dari suku pertama yang dikendalikan oleh $\alpha(s)$ dan suku kedua yang dikendalikan oleh $\beta(s)$. Suku pertama mengontrol elastisitas (*elasticity*) dan suku kedua mengontrol kekakuan (*stiffness*) snake. Untuk penyederhanaan, bobot $\alpha(s)$ dan $\beta(s)$ **diasumsikan seragam**, sehingga $\alpha(s) = \alpha$ dan $\beta(s) = \beta$ hal ini mencegah agar deformabilitas snake tidak membentuk sudut (Ivins and Porrill, 1995). Tidak ada aturan dalam menentukan α dan β, tetapi yang perlu diperhatikan adalah semakin kecil nilai α mengakibatkan jarak tiap titik pada kurva semakin tidak teratur dan sebaliknya semakin besar nilai α mengakibatkan jarak tiap titik pada kurva semakin teratur. Untuk parameter β, semakin kecil nilainya akan menyebabkan bentuk kurva menjadi semakin tidak *smooth* atau dapat membentuk sudut dan sebaliknya semakin besar nilai β menyebabkan kurva semakin *smooth* (Ickhsan, 2020).

2.4.3 Energi eksternal

Energi internal berasal dari kurva *snake*, sedangkan energi eksternal berasal dari luar yakni dari citra itu sendiri.

Jika diberikan citra tingkat abu-abu (*gray-level image*) $I(x, y)$, maka fungsi energi eksternal yang sesuai meliputi (Xu and Prince, 1998):

$$E_{ext}^{(1)}(x, y) = -|\nabla I(x, y)|^2 \tag{2.12}$$

$$E_{ext}^{(2)}(x, y) = -|\nabla \left[G_\sigma(x, y) * I(x, y)\right]|^2 \tag{2.13}$$

Jika citra tersebut adalah citra biner (*black-white*), energi eksternal yang sesuai di antaranya adalah sebagai berikut:

$$E_{ext}^{(3)}(x, y) = I(x, y) \tag{2.14}$$

$$E_{ext}^{(4)}(x, y) = G_\sigma(x, y) * I(x, y) \tag{2.15}$$

di mana $G_\sigma(x, y)$ adalah fungsi *Gaussian* dengan standar deviasi σ, ∇ adalah operator gradien. $*$ mewakili konvolusi $G_\sigma(x, y) * I(x, y)$, di mana $I(x, y)$ adalah fungsi intensitas citra. ∇ adalah operator gradien, kita dapat menggunakan operator Sobel pada persamaan (2.7) dan (2.8) untuk menjalankan fungsi energi eksternal ini.

Substitusi Energi internal (2.11) dan Energi Eksternal (2.12) - (2.15) ke fungsional energi *snake* (2.10), menghasilkan persamaan energi *snake* menjadi:

$$E_{snake} = \int_0^1 \left(\frac{1}{2} \left(\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2 \right) + E_{ext}^{(i)}(x, y) \right) ds. \tag{2.16}$$

dengan $i = 1, 2, 3, 4$.

2.5 *Active contour evolution*

Kurva *snake* akan bergerak menuju batas dari *feature* dari sebuah citra, proses ini disebut *active contour evolution*. Perhitungan evolusi *snake* yang meminimalkan persamaan energi (2.16) ketika $\alpha(s) = \alpha$ dan $\beta(s) = \beta$, harus memenuhi persamaan Euler berikut (Xu and Prince, 1998):

$$\alpha \mathbf{v}_{ss}(s) - \beta \mathbf{v}_{ssss}(s) + \nabla E_{ext}^{(i)} = 0 \tag{2.17}$$

Untuk menemukan solusi persamaan (2.17), kurva snake **v** dibuat dinamis terhadap waktu t. Turunan parsial **v** terhadap t ditetapkan sama dengan ruas kiri persamaan (2.17) sebagai berikut (Xu and Prince, 1998):

$$\mathbf{v}_t(s,t) = \alpha \mathbf{v}_{ss}(s,t) - \beta \mathbf{v}_{ssss}(s,t) + \nabla E_{ext}^{(i)} = 0 \tag{2.18}$$

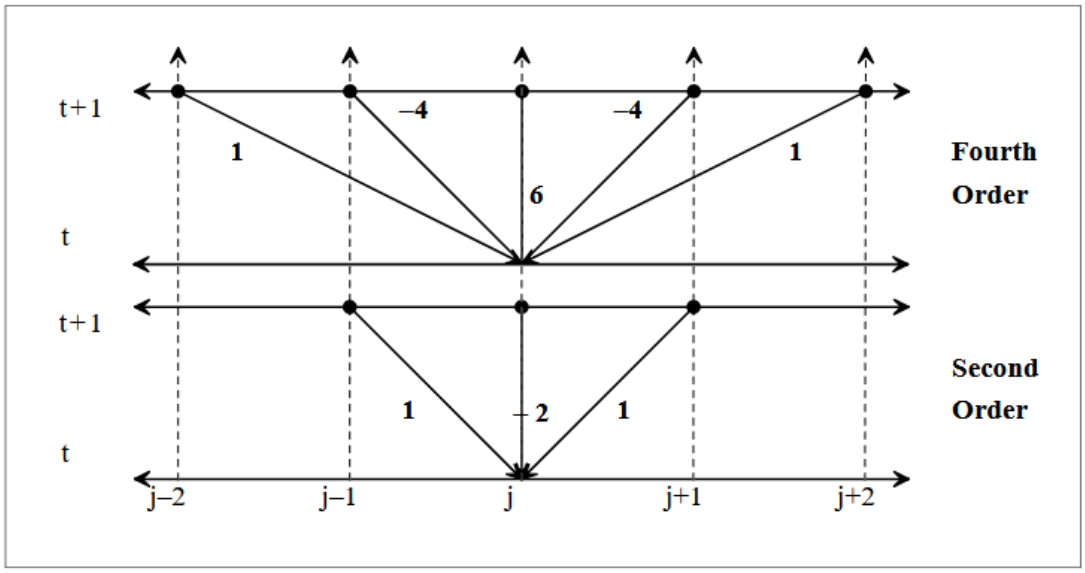
atau dapat ditulis sebagai berikut (Ivins and Porrill, 1995):

$$\frac{\partial \mathbf{v}}{\partial t} = \alpha \frac{\partial^2 \mathbf{v}}{\partial s^2} - \beta \frac{\partial^4 \mathbf{v}}{\partial s^4} + \frac{\partial E_{ext}^{(i)}}{\partial \mathbf{v}} = 0 \tag{2.19}$$

v pada persmaan (2.19) dapat dipisahkan menjadi komponen/elemen x dan y, katakanlah u_j adalah komponen/elemen *snake* **v** di mana $j = 0, 1, .., N-1$ sebagai aproksimasi diskrit untuk $x(s)$ dan $y(s)$, dan t melambangkan iterasi, maka persamaan (2.19) menjadi:

$$\frac{\partial u_j^t}{\partial t} = \alpha \frac{\partial^2 u_j^t}{\partial s^2} - \beta \frac{\partial^4 u_j^t}{\partial s^4} + \frac{\partial E_{ext}^{(i)}}{\partial u_j^t} = 0 \tag{2.20}$$

Turunan pada persamaan (2.20) dapat diaproksimasikan menggunakan *finite differences* yang ditunjukan pada gambar berikut:



Gambar 2.8: Aproksimasi turunan dengan *finite differences* (Ivins and Porrill, 1995)

Turunan $\frac{\partial u_j^t}{\partial t}$ diaproksimasikan menjadi $\frac{u_j^{t+1} - u_j^t}{\delta t}$ (Ivins and Porrill, 1995). Turunan orde kedua, yaitu $\frac{\partial^2 u_j^t}{\partial s^2}$ diaproksimasikan menjadi $\frac{(u_{j+1}^{t+1} + u_{j-1}^{t+1} - 2u_j^{t+1})}{\delta s^2}$, sedangkan turunan orde keempat, yaitu $\frac{\partial^4 u_j^t}{\partial s^4}$ diaproksimasikan menjadi $\frac{(u_{j+2}^{t+1} - 4u_{j+1}^{t+1} + 6u_j^{t+1} - 4u_{j-1}^{t+1} + u_{j-2}^{t+1})}{\delta s^4}$. Turunan dalam persamaan (2.20) yang diaproksimasi menggunakan *finite differences* seperti yang ditunjukkan pada Gambar (2.8) adalah sebagai berikut:

$$\frac{u_j^{t+1} - u_j^t}{\delta t} = \frac{\alpha}{\delta s^2}(u_{j+1}^{t+1} + u_{j-1}^{t+1} - 2u_j^{t+1}) - \frac{\beta}{\delta s^4}(u_{j+2}^{t+1} - 4u_{j+1}^{t+1} + 6u_j^{t+1} - 4u_{j-1}^{t+1} + u_{j-2}^{t+1}) + \frac{\partial E_{ext}^{(i)}}{\partial u_j^t} \quad (2.21)$$

di mana t dan $t+1$ mewakili parameter waktu yang berurutan dengan langkah waktu (*time step*) δt. δs mewakili ukuran langkah (*step size*). Mememindahkan δt ke

ruas kanan persamaan membuat persamaan (2.21) menjadi:

$$bu_{j+2}^{t+1} - (a+4b)u_{j+1}^{t+1} + (1+2a+6b)u_j^{t+1} - (a+4b)u_{j-1}^{t+1} + bu_{j-2}^{t+1} = u_j^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_j^t} \quad (2.22)$$

di mana $a \equiv \alpha \frac{\delta t}{\delta s^2}$ dan $b \equiv \beta \frac{\delta t}{\delta s^4}$. Persamaan (2.22) dapat juga ditulis sebagai berikut:

$$pu_{j+2}^{t+1} + qu_{j+1}^{t+1} + ru_j^{t+1} + qu_{j-1}^{t+1} + pu_{j-2}^{t+1} = \tilde{u}_j^{t+1} \quad (2.23)$$

di mana $p \equiv b$, $q \equiv -a-4b$, dan $r \equiv 1+2a+6b$, sedangkan $\tilde{u}_j^{t+1} = u_j^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_j^t}$. Persamaan (2.23) ini dapat ditulis dalam bentuk matriks untuk koordinat x dan y dari setiap elemen kurva *snake* **v**.

$$\begin{bmatrix} r & q & p & & & p & q \\ q & r & q & p & & & p \\ p & q & r & q & p & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & p & q & r & q & p \\ p & & & p & q & r & q \\ q & p & & & p & q & r \end{bmatrix} \begin{bmatrix} u_0^{t+1} \\ u_1^{t+1} \\ u_2^{t+1} \\ \vdots \\ u_{N-3}^{t+1} \\ u_{N-2}^{t+1} \\ u_{N-1}^{t+1} \end{bmatrix} = \begin{bmatrix} \tilde{u}_0^{t+1} \\ \tilde{u}_1^{t+1} \\ \tilde{u}_2^{t+1} \\ \vdots \\ \tilde{u}_{N-3}^{t+1} \\ \tilde{u}_{N-2}^{t+1} \\ \tilde{u}_{N-1}^{t+1} \end{bmatrix} \quad (2.24)$$

di mana

$$\mathbf{M} = \begin{bmatrix} r & q & p & & & p & q \\ q & r & q & p & & & p \\ p & q & r & q & p & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & p & q & r & q & p \\ p & & & p & q & r & q \\ q & p & & & p & q & r \end{bmatrix} \tag{2.25}$$

$$\mathbf{u}_j^{t+1} = \begin{bmatrix} u_0^{t+1} \\ u_1^{t+1} \\ u_2^{t+1} \\ \vdots \\ u_{N-3}^{t+1} \\ u_{N-2}^{t+1} \\ u_{N-1}^{t+1} \end{bmatrix} \tag{2.26}$$

$$\tilde{\mathbf{u}}_j^{t+1} = \begin{bmatrix} \tilde{u}_0^{t+1} \\ \tilde{u}_1^{t+1} \\ \tilde{u}_2^{t+1} \\ \vdots \\ \tilde{u}_{N-3}^{t+1} \\ \tilde{u}_{N-2}^{t+1} \\ \tilde{u}_{N-1}^{t+1} \end{bmatrix} = \begin{bmatrix} u_0^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_0^t} \\ u_1^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_1^t} \\ u_2^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_2^t} \\ \vdots \\ u_{N-3}^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_{N-3}^t} \\ u_{N-2}^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_{N-2}^t} \\ u_{N-1}^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial u_{N-1}^t} \end{bmatrix} \tag{2.27}$$

$$p \equiv \beta \frac{\delta t}{\delta s^4}, q \equiv -\alpha \frac{\delta t}{\delta s^2} - 4\beta \frac{\delta t}{\delta s^4}, r \equiv 1 + 2\alpha \frac{\delta t}{\delta s^2} + 6\beta \frac{\delta t}{\delta s^4} \tag{2.28}$$

Mengalikan kedua sisi dari persamaan (2.24) dengan inverse dari matriks **M** memberikan solusi akhir persamaan *snake* menjadi:

$$\mathbf{u}_j^{t+1} = \mathbf{M}^{-1}\left(\mathbf{u}_j^t + \delta t \frac{\partial E_{ext}^{(i)}}{\partial \mathbf{u}_j^t}\right) \tag{2.29}$$

Perlu diingat **u** adalah komponen/elemen x dan y pada kurva *snake*, komponen/elemen ini dapat dinyatakan dengan vektor **x** dan **y**. Jika $E_{ext}^{(i)}$ dinotasikan menjadi fungsi **f** maka :

$$\frac{\partial E_{ext}^{(i)}}{\partial \mathbf{u}_j^t} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}_j^t} \tag{2.30}$$

Hal ini membuat persamaan (2.29) dapat ditulis sebagai berikut:

$$\mathbf{x}_j^{t+1} = \mathbf{M}^{-1}\left(\mathbf{x}_j^t + \delta t\ \mathbf{f}_x\right) \tag{2.31}$$

dan

$$\mathbf{y}_j^{t+1} = \mathbf{M}^{-1}\left(\mathbf{y}_j^t + \delta t\ \mathbf{f}_y\right) \tag{2.32}$$

$\mathbf{f}_x$ dan $\mathbf{f}_y$ adalah vektor-vektor yang bersesuaian dengan bentuk turunan pertama $\frac{\partial \mathbf{f}}{\partial \mathbf{u}_j^t}$ sebagai berikut:

$$\mathbf{f}_x = f_x(x_j^t, y_j^t) \tag{2.33}$$

$$\mathbf{f}_y = f_y(x_j^t, y_j^t) \tag{2.34}$$

di mana x_j^t dan y_j^t adalah pasangan koordinat dari kurva *snake*. $\mathbf{f}_x$ dan $\mathbf{f}_y$ dapat dicari menggunakan teori gradien citra.

2.6 Interpolasi Bilinear

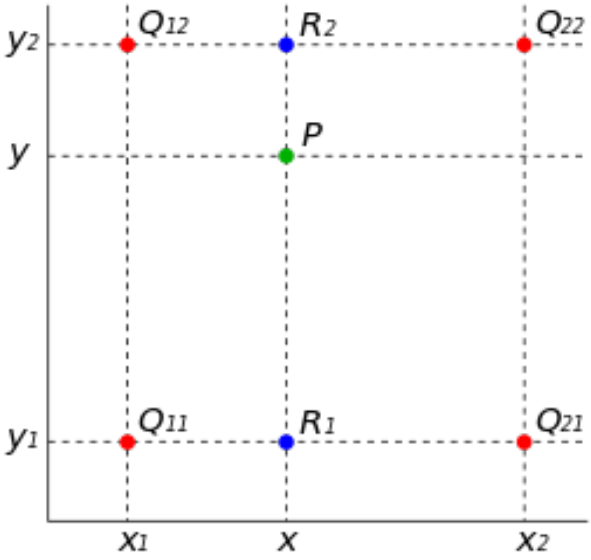
Interpolasi Bilinear adalah metode interpolasi fungsi dari dua variabel (misal x dan y). Metode ini biasanya digunakan untuk mengaproksimasi nilai data di antara beberapa titik dalam *2D rectangular grid*, misalnya di dalam matriks (William H. Press and Flannery).

Misal kita ingin mencari nilai fungsi f yang tidak diketahui di titik (x, y) dan diasumsikan kita mengetahui nilai f pada beberapa empat titik $Q_11 = (x_1, y_1), Q_12 = (x_1, y_2), Q_21 = (x_2, y_1), danQ_22 = (x_2, y_2)$ maka langkah yang harus dilakukan sebagai berikut:

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x1} f(Q_11) + \frac{x - x_1}{x_2 - x_1} f(Q_21) = R1 \tag{2.35}$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x1} f(Q_12) + \frac{x - x_1}{x_2 - x_1} f(Q_22) = R2 \tag{2.36}$$

$$f(x, y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) = P \tag{2.37}$$

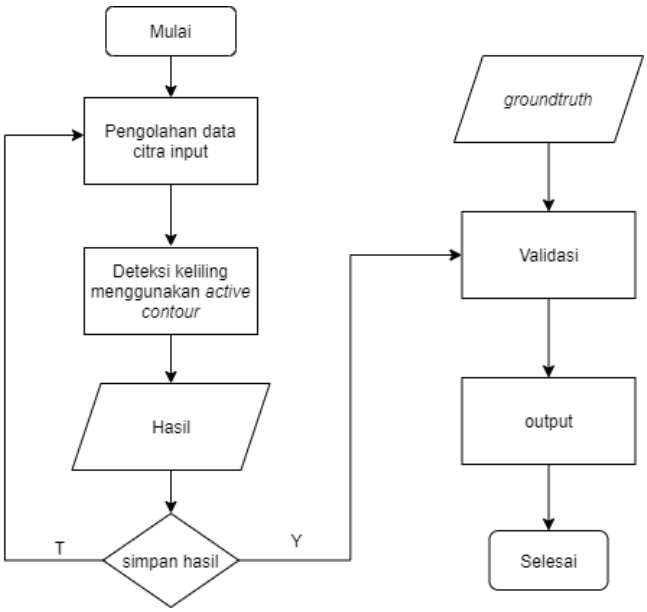


Gambar 2.9: interpolasi bilinear

BAB III

METODOLOGI PENELITIAN

Berikut ini diagram alir penelitian yang akan dilakukan:



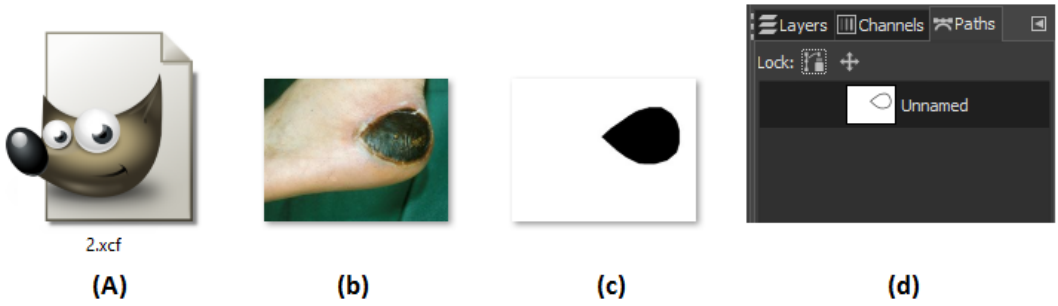
Gambar 3.1: Diagram alir penelitian

3.1 Pengolahan data citra input

Sampel yang baik adalah sampel yang mencerminkan populasinya (Amirullah, 2015). Permata menggunakan data sebanyak 20 citra preparat darah pada penelitiannya tentang segmentasi parasit malaria menggunakan *snake*(Permata, 2015), lalu pada penelitian lain Fadillah menggunakan data sebanyak 15 citra CT-Scan paru-paru pada penelitiannya tentang segmentasi citra paru-paru menggunakan *snake*, dan Constantia menggunakan 21 data citra sapi dalam penelitiannya tentang estimasi bobot sapi menggunakan *snake*(Constantia et al., 2019).

Dataset luka yang penulis dapat berjumlah 108, sebanyak 37 data tidak dapat

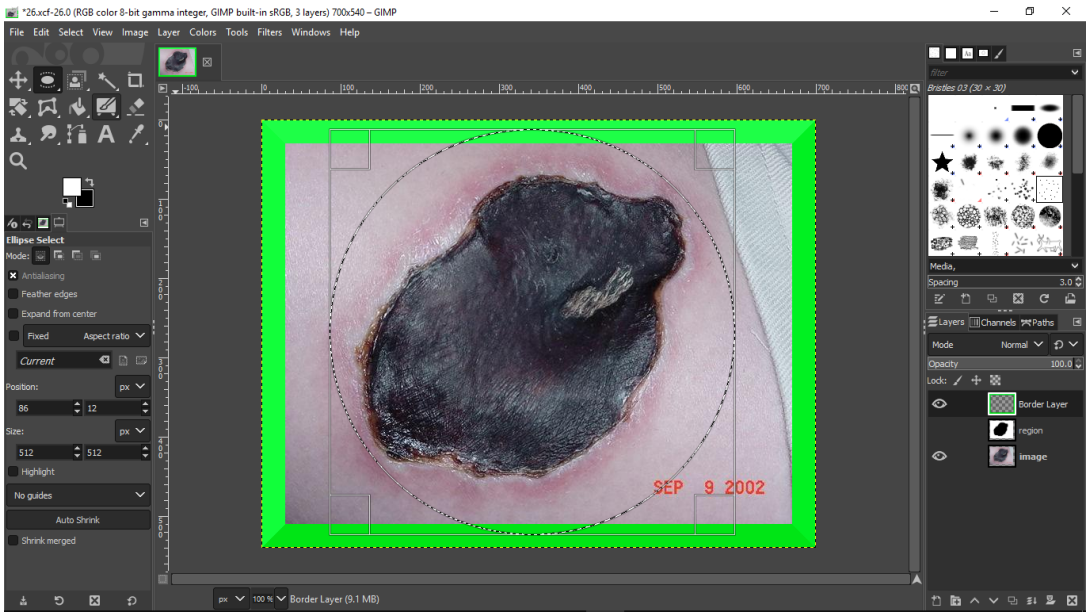
dipakai karena data tersebut memiliki duplikasi dengan data lain sehingga data yang tersedia berjumlah 71 buah citra luka yang penulis jadikan sebagai populasi sekaligus sebagai sampel penelitian (sampel = populasi) dengan kategori luka hitam sebanyak 24 citra, luka kuning sebanyak 15 citra, dan luka merah sebanyak 32 citra. *Dataset* ini didapat dari penelitian luka Ns. Ratna Aryani, M.Kep, tahun 2018 (Ratna Aryani, 2018) yang tersedia di *repository* `https://github.com/mekas/InjuryDetection`. Data yang penulis gunakan adalah data-data yang berekstensi .xcf yang dapat dibuka dengan *software* GIMP, pemrosesan data sebelum deteksi menggunakan *snake* dan GVF dilakukan menggunakan *software* GIMP. *Dataset* ini masing-masing di dalamnya terdapat *layer* citra (luka), *layer* region (luka), dan *path* sebagai berikut :



Gambar 3.2: (a) Data citra format .xcf, (b) *layer* citra (luka), (c) *layer* region, (d) *path*

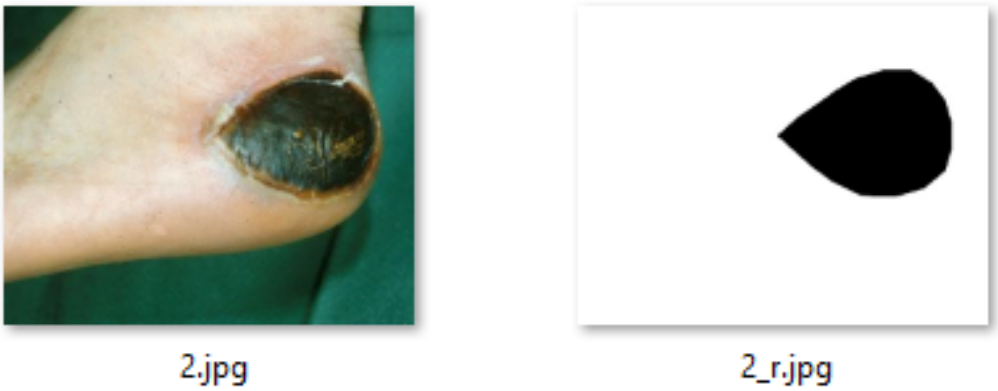
langkah selanjutnya adalah mengubah ukuran (*resize*) citra *size* yang besar ke ukuran yang lebih kecil agar proses deteksi menjadi lebih cepat. Penulis mengubah ukuran citra menggunakan fitur *rescale image* sehingga ukuran citra tidak lebih dari 2 *megabyte*. Kemudian penulis mengecek masing-masing citra dengan fitur *eclipse select* untuk mengetahui apakah objek luka pada data citra pas berada di dalam lingkaran yang akan dijadikan sebagai inisialisasi awal. Ukuran lingkaran tidak boleh lebih besar dari ukuran citra. Jika ukuran lingkaran lebih besar daripada ukuran citra, maka perlu ditambahkan *border* yang dibuat menggunakan fitur *add*

border sebagai berikut :



Gambar 3.3: Citra luka yang telah dicek menggunakan fitur *eclipse select* dan ditambahkan *border*

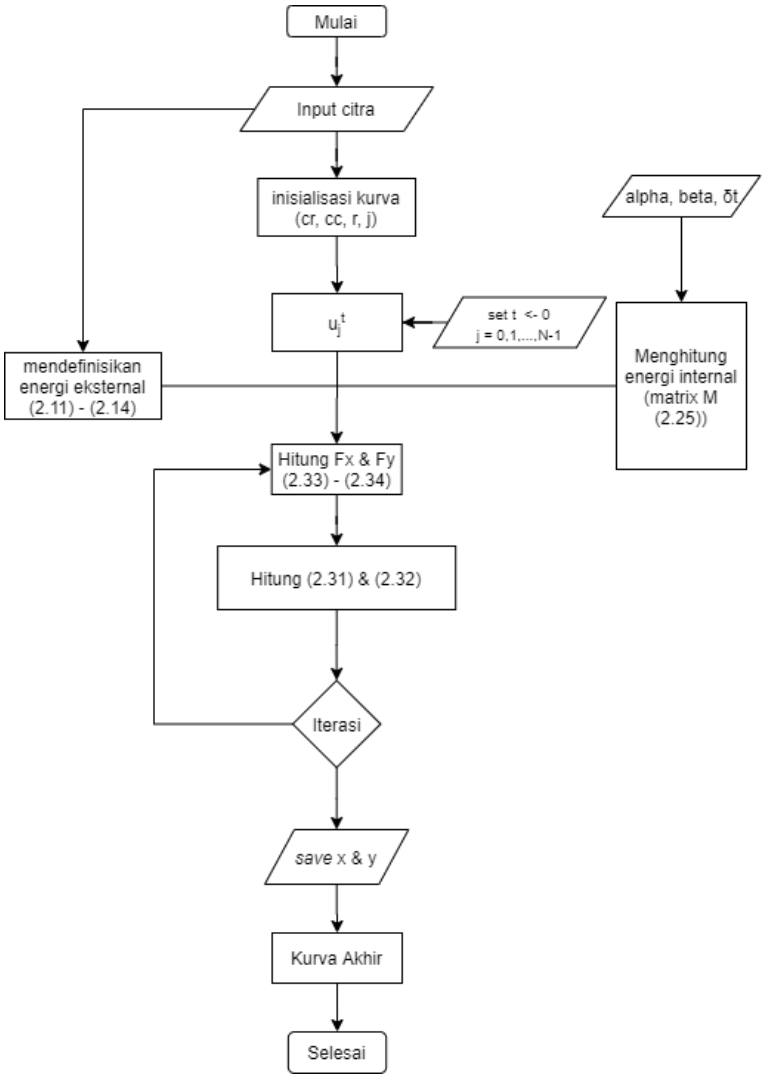
Setelah proses *resize* sampai pengecekan lingkaran, selanjutnya penulis *export layer* masing masing ke format .jpg.



Gambar 3.4: Citra luka dan region luka

3.2 Deteksi keliling menggunakan *Active Contour* (*snake*)

Langkah selanjutnya adalah deteksi keliling menggunakan *snake*. Langkah-langkahnya adalah sebagai berikut:



Gambar 3.5: Diagram alir *snake*

3.2.1 Inisialisasi kurva awal *Active Contour*

Sebelum menjalankan *Active Contour (snake)* untuk mendeteksi keliling luka, hal yang harus dilakukan adalah menginisialisasi kurva awal. Penulis

mendefinisikan kurva awal *snake* berbentuk lingkaran dan diinisialisasikan secara manual di atas citra. Penulis memberikan anotasi pada masing-masing data di mana cr (*center_row*) dan cc (*center_column*) menunjukkan titik pusat lingkaran pada citra, dan r adalah jari-jari citra sebagai berikut:

Gambar 3.6: (A) citra luka, (b) citra luka (*grayscale*), (c) inisialisasi kurva awal dengan cr = 120 , cc = 265, r = 85

3.3 *Ground truth*

Setelah mendapatkan data citra dan region selanjutnya adalah mencari *ground truth* dari masing masing citra. Penulis menggunakan fitur *stroke path* dari GIMP untuk mendapatkan tepi dari data region yang akan dijadikan sebagai *ground truth*. Hal ini bertujuan untuk keperluan menampilkan hasil kurva akhir dan di letakan di atas *ground truth*

Gambar 3.7: Komparasi Citra luka, region luka, dan *groundtruth*

3.4 Validasi

Validasi yang penulis gunakan adalah dengan cara menghitung selisih piksel dari area kurva akhir *snake* dengan area *ground truth* sehingga mendapatkan nilai akurasi sebagai berikut:

$$Akurasi(\%) = 100 - \left| \frac{(luas\ area\ groundtruth - luas\ area\ kurva\ akhir)}{luas\ area\ groundtruth} * 100 \right| \tag{3.1}$$

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengolahan data citra input

Langkah pertama dalam pengolahan data citra input adalah Proses konversi citra luka RGB menjadi citra luka *grayscale*. Proses konversi citra luka menjadi citra luka *grayscale* dilakukan menggunakan bahasa pemrograman *python* dengan bantuan *library scikit-image*, *library* ini dipilih karena fungsi yang digunakan sama dengan metode *luminosity* di persamaan 2.1 (rgb). berikut *source code* nya:

```
img = imread("some_image.jpg")
img_gray = rgb2gray(img)
```

Gambar 4.1: *source code* konversi citra RGB ke *grayscale*

fungsi `imread` dan `rgb2gray` di-*import* dari *library* `skimage.io` dan `skimage.color` yang masing-masing fungsinya adalah untuk membaca dan melakukan konversi citra RGB ke citra *grayscale*. Data citra *grayscale* disimpan kedalam variabel `img_gray`. Rincian hasil dari proses ini tersedia di bagian **lampiran B**.

4.2 Deteksi keliling luka menggunakan *active contour* (*snake*)

Proses deteksi keliling luka menggunakan *snake* versi asli *integer* dan metode *snake* yang ditambahkan interpolasi yang datanya berupa nilai *float*. Berikut tahapannya:

4.2.1 Inisialisasi kurva awal

Rincian hasil dari proses ini tersedia di bagian *lampiran C dan D*. Berikut *source code* yang bekerja dengan baik untuk inisialisasi kurva awal:

```
theta = np.linspace(0, 2*np.pi, num_of_sample)
r = cr + rad*np.sin(theta)
c = cc + rad*np.cos(s)
snake_init = np.array([r, c]).T

snake_xy = snake_init[:, ::-1].astype(int)
x = snake_xy[:, 0]
y = snake_xy[:, 1]
```

Gambar 4.2: *source code* inisialisasi kurva lingkaran versi *integer*

```
theta = np.linspace(0, 2*np.pi, num_of_sample)
r = cr + rad*np.sin(theta)
c = cc + rad*np.cos(theta)
snake = np.array([r, c]).T

snake_xy = snake[:, ::-1]
x = snake_xy[:, 0].astype(float)
y = snake_xy[:, 1].astype(float)
```

Gambar 4.3: *source code* inisialisasi kurva lingkaran versi interpolasi

Potongan kode (4.2) dan (4.3) diimplementasi berdasarkan algoritma pembuatan lingkaran menggunakan bentuk parametrik (*parametric form*) lingkaran (2.7). `theta` yang didapatkan menggunakan fungsi `np.linspace` milik `numpy` menghasilkan *array* dengan *step* yang sama dan nilai-nilai di dalam `theta` berjumlah `num_of_sample` dengan *range* 0 sampai 2π, `cc` dan `cr` adalah titik tengah lingkaran dan`rad` adalah jari-jari. Kurva awal *snake* ini disimpan kedalam variabel `snake_xy`.

4.2.2 Energi internal

output dari energi internal adalah sebuah matriks yang elemennya dipengaruhi oleh parameter α, β, dan *time step* seperti pada matriks (2.25). Berikut *source code* yang bekerja dengan baik untuk menghasilkan matriks energi internal:

```
a = beta
b = -(4*beta + alpha)
c = 6*beta + 2*alpha

eye_n = np.eye(n, dtype=float)
c_axis = c * eye_n
b_axis = b * ( np.roll(eye_n, -1, axis=0) + np.roll(eye_n, -1, axis=1) )
a_axis = a * ( np.roll(eye_n, -2, axis=0) + np.roll(eye_n, -2, axis=1) )
A = c_axis + b_axis + a_axis

inv = np.linalg.inv(A + gamma * eye_n)
```

Gambar 4.4: *source code* untuk menghasilkan energi internal

Penulis mengimplementasi (2.25) dengan memanggil beberapa fungsi dari *library numpy*. Fungsi `np.eye(n)` menghasilkan matriks identitas dengan ukuran `n x n`. `np.roll` digunakan untuk mengubah nilai pada diagonal tertentu pada suatu matriks. `np.linalg.inv` digunakan untuk menghitung invers matriks. Variabel `a` dan `b` masing masing mewakili p dan q pada matriks (2.25). `eye_n` adalah matriks identitas dengan ukuran `n x n` di mana `n` adalah panjang dari vektor kurva *snake*. `a_axis`, `b_axis`, dan `c_axis` masing-masing adalah elemen matriks yang mengisi p, q, dan r pada matriks (2.25). `A + gamma * eye_n` adalah cara untuk membuat elemen-elemen pada matriks sesuai dengan p, q, dan r pada matriks (2.25). Matriks energi internal kemudian dihitung inversnya dan invers dari matriks tersebut disimpan kedalam variabel `inv` untuk digunakan pada persamaan update kurva *snake*. Parameter α, β, dan *time step* untuk setiap data tersedia di bagian *lampiran C dan D*

4.2.3 Energi eksternal

Untuk versi *integer* energi eksternal menggunakan persamaan (2.13). Untuk versi interpolasi yang dipilih adalah persamaan 2.15, kemudian persamaan 2.15 diproses lagi menggunakan gradien Sobel untuk mendapatkan peta tepi dari citra luka. Berikut *source code* yang bekerja dengan baik untuk menghasilkan energi eksternal:

```
ext = gaussian(im_gray, sigma)
ext = sobel(ext)
ext = -ext**2
```

Gambar 4.5: *source code* energi eksternal untuk *snake* versi *integer*

```
ext = gaussian(im_gray, sigma)
ext = img_as_float(ext)
ext = ext.astype(float, copy=False)
ext = sobel(ext)
```

Gambar 4.6: *source code* energi eksternal untuk *snake* versi interpolasi

Penulis mengimplementasi energi eksternal dengan memanggil beberapa fungsi dari *library scikit-image*. Fungsi `gaussian` menghasilkan citra gaussian $G_\sigma(x, y) * I(x, y)$ dengan standar deviasi `sigma`, sedangkan yang dihasilkan fungsi `sobel` adalah besar gradien sobel, sesuai dengan persamaan (2.9). Energi eksternal disimpan kedalam variabel `ext`. Rincian hasil dari proses ini tersedia di bagian *lampiran C dan D*.

4.2.4 Proses update intersi kurva

Berikut *source code* yang bekerja dengan baik untuk proses update iterasi kurva atau dapat disebut *snake evolution*:

```
# potential force
gy, gx = np.gradient(ext)

xt = np.copy(x)
yt = np.copy(y)

max_num_iter=n
for i in range(max_iter):
        fx = np.array([])
        fy = np.array([])

        for i in range(n):
                fx = np.append(fx, gx[yt[i]][xt[i]])
                fy = np.append(fy, gy[yt[i]][xt[i]])

        xn = np.dot(inv, xt + gamma * fx) # acton, ivins equation
        yn = np.dot(inv, yt + gamma * fy)

        xt = np.round(xn).astype(int)
        yt = np.round(yn).astype(int)

snake_final = np.array([xt, yt]).T
```

Gambar 4.7: *source code* proses update iterasi kurva versi *integer*

```
# potential force
gy, gx = np.gradient(ext)

# Interpolate for smoothness
intp1 = RectBivariateSpline(np.arange(gx.shape[1]),
                                                    np.arange(gx.shape[0]),
                                                    gx.T, kx=2, ky=2, s=0)

intp2 = RectBivariateSpline(np.arange(gy.shape[1]),
                                                    np.arange(gy.shape[0]),
                                                    gy.T, kx=2, ky=2, s=0)


# deform snake
max_num_iter=n
max_px_move=1.0

xt = np.copy(x)
yt = np.copy(y)

for i in range(max_num_iter):
        # fx & fy
        fx = intp1(xt, yt, dx=0, grid=False).astype(float, copy=False)
        fy = intp2(xt, yt, dy=0, grid=False).astype(float, copy=False)

        # skimage equation
        xn = np.dot(inv, gamma * xt + fx)
        yn = np.dot(inv, gamma * yt + fy)

        # Movements are capped to max_px_move per iteration. skimage
        dx = max_px_move * np.tanh(xn - xt)
        dy = max_px_move * np.tanh(yn - yt)

        xt += dx
        yt += dy

snake_final = np.array([xt, yt]).T
```

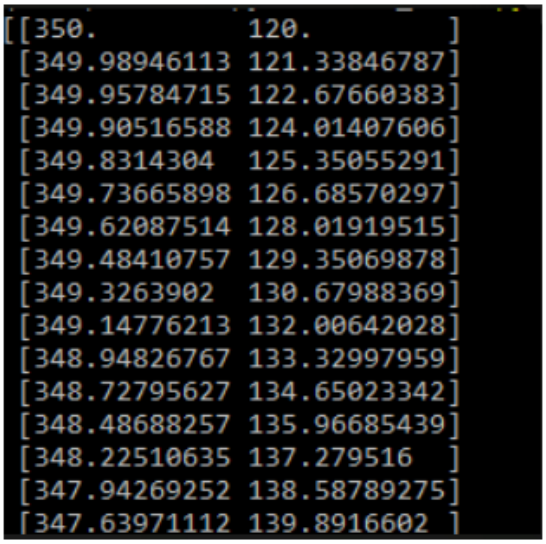
Gambar 4.8: *source code* proses update iterasi kurva versi interpolasi

Metode *snake* versi *integer* dan interpolasi memakai metode gradien arah *direction* untuk mendapatkan turunan pertama citra yang akan digunakan untuk persamaan (2.33) dan (2.34). Gradien *direction* x dan y didapat dengan memanggil `np.gradient` dan disimpan kedalam variabel `gx` dan `gy`.

Proses iterasi dilakukan sebanyak `max_num_iter` kali dan yang dilakukan adalah mengupdate nilai `fx` dan `fy`. Untuk versi *integer* nilai `fx` dan `fy` dapat

dihitung secara langsung karena nilainya tersedia di dalam `gx` dan `gy` sedangkan untuk versi interpolasi nilainya harus didapatkan melalui proses interpolasi.

Bentuk kurva *snake* untuk setiap iterasi terdiri dari pasangan koordinat dengan nilai *float*, hal ini menimbulkan masalah pada operasi *finding contour* di setiap iterasi yang menghasilkan koordinat bilangan *float*. Kurva harus diletakkan di atas citra yang menyebabkan terjadi *missing value* pada kurva jika nilai koordinatnya terdiri dari nilai *float*.

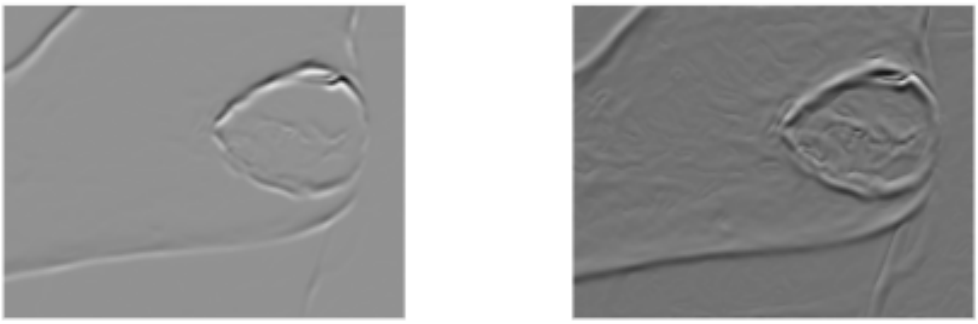


Gambar 4.9: Bentuk kurva *snake* terdiri dari pasangan koordinat dengan nilai *float*

Untuk mengatasi masalah ini, pada *snake* versi integer dilakukan pembulatan (integer) terhadap kurva. Selanjutnya untuk versi interpolasi, peneliti melakukan *tracing* terhadap metode *snake* yang ada di *library scikit-image* dan menemukan sesuatu pada *library* tersebut terdapat modifikasi dari *snake* integer dengan menambahkan interpolasi, sisanya dihipotesiskan sebagai *code minor*. Kontribusi terbesar pada penelitian ini terdapat pada penemuan bahwa interpolasi dibutuhkan untuk menjalankan *snake*.

Proses interpolasi yang dilakukan *library scikit-image* adalah dengan memanggil *Class* `RectBivariateSpline` (acm). *Class* ini menggunakan

pendekatan *bivariate spline* di atas *rectangular mesh*, dalam hal ini adalah citra (matriks). `RectBivariateSpline` dapat digunakan untuk interpolasi data (rbs). Proses interpolasi yang dilakukan *library scikit-image* dapat dianggap dalam langkah *preprocessing* terhadap citra asli yang mana pengaruhnya adalah arah tepinya semakin jelas dibanding dengan citra asli.



Gambar 4.10: komparasi gradien citra arah (*direction*) y (`gy`). (a) citra asli (b) citra hasil interpolasi

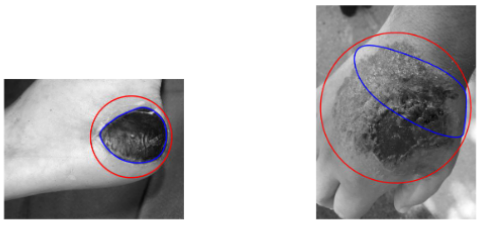
Proses untuk mendapatkan nilai `fx` dan `fy` pada *snake* versi interpolasi adalah dengan menggunakan *class* `RectBivariateSpline` milik `scipy` (rbs). Dengan memanggil metode `RectBivariateSpline` kita dapat mendapatkan nilai `fx` dan `fy` terhadap elemen kurva yang berupa bilangan *float*. Sayangnya pada saat melakukan penelitian penulis tidak menemukan algoritma bagaimana cara mendapatkan nilai `fx` dan `fy` menggunakan `RectBivariateSpline` karena *source code* milik *scipy* sangat kompleks.

Selanjutnya untuk proses update kurva versi *integer* perhitungannya dapat menggunakan persamaan (2.31) dan (2.32), sedangkan pada versi interpolasi penulis menggunakan persamaan milik *skimage* (acm) karena jika menggunakan persamaan (2.31) dan (2.32) kurva tidak mau bergerak. Persamaan ini memodifikasi sedikit dari

persamaan *snake* asli dengan penambahan kode untuk membatasi pergerakan kurva tidak lebih dari `max_px_move`. Setelah selesai mengupdate kurva, kurva akhir disimpan kedalam variabel `snake_final`.

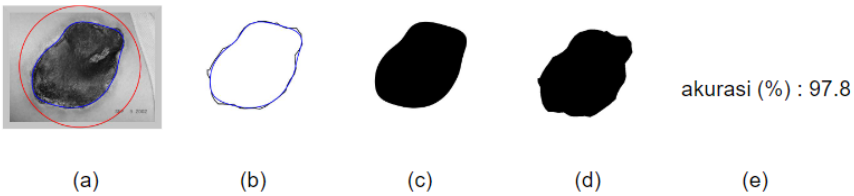
4.3 Eksperimen

Source code eksperimen disimpan ke dalam repository yang dapat diakses di `https://github.com/m-rizki/skripsi` di bawah lisensi *GNU General Public License v3.0*. Rancangan eksperimen deteksi keliling luka menggunakan *snake* dan *snake* yang ditambahkan interpolasi dilakukan dengan menjalankan program *snake* hingga kurva akhir menutupi objek luka pada citra untuk semua data (populasi) yang tersedia. Proses ini memerlukan penyesuaian parameter algoritma *snake* untuk setiap citra. Dari hasil yang didapatkan terdapat dua kategori hasil deteksi, yang pertama adalah kurva hasil deteksi berhasil menutupi luka (berhasil) dan kurva hasil deteksi tidak berhasil menutupi luka (gagal). Rincian parameter dan hasil dari proses deteksi tersedia di bagian *lampiran C dan D*



Gambar 4.11: (a) Berhasil (b) Gagal

Setelah hasil dikateorikan, langkah selanjutnya adalah mengidentifikasi area kurva hasil deteksi *snake* yang berhasil dan di bandingkan terhadap area *ground truth* untuk kemudian dihitung akurasinya menggunakan persamaan (3.1). Rincian hasil dari proses ini tersedia di bagian *lampiran C dan D*



Gambar 4.12: (a) Hasil deteksi (b) Hasil deteksi *overlay* dengan *ground truth* (c) Area kurva akhir (d) Area *ground truth* (e) Akurasi

4.4 Analisa hasil

Berikut analisa hasil setelah melakukan eksperimen terhadap populasi:

1. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* kategori luka hitam menghasilkan 5 data (dari 24 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 84.2%.

2. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* kategori luka kuning menghasilkan 5 data (dari 15 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 76.14%.

3. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* kategori luka merah menghasilkan 2 data (dari 30 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 62.25%.

4. Hasil dari deteksi keliling luka menggunakan *snake* versi *intepolasi* kategori luka hitam menghasilkan 21 data (dari 24 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 87.6%.

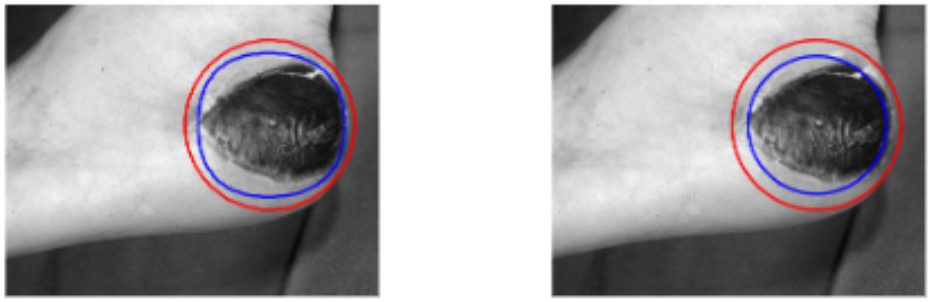
5. Hasil dari deteksi keliling luka menggunakan *snake* versi *interpolasi* kategori luka kuning menghasilkan 11 data (dari 15 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 79.2%.

6. Hasil dari deteksi keliling luka menggunakan *snake* versi *interpolasi* kategori luka merah menghasilkan 12 data (dari 30 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 89.7%.

7. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* untuk semua kategori menghasilkan 12 data (dari 71 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 77.18%.

8. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* untuk semua kategori menghasilkan 44 data (dari 71 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 86.1%.

Sebagai tambahan, setelah dicek kembali ternyata terdapat kesalahan pada saat proses update iterasi kurva untuk metode *snake* versi *integer*. Sebelumnya penulis melakukan pembulatan terhadap kurva di luar iterasi dan di dalam iterasi, tetapi ternyata seharusnya penulis cukup melakukan proses pembulatan pada saat perhitungan `fx` dan `fy` di dalam iterasi saja sehingga kurva akhirnya tetap terdiri dari koordinat yang nilainya *float*. Jika dibandingkan hasilnya sebagai berikut.



(a) (b)

Gambar 4.13: (a) Hasil *snake* versi integer dengan pembulatan kurva di luar dan di dalam iterasi (b) Hasil *snake* versi integer dengan pembulatan kurva hanya di dalam iterasi

```
# potential force
gy, gx = np.gradient(ext)

xt = np.copy(x)
yt = np.copy(y)

max_num_iter=n
for i in range(max_iter):
fx = np.array([])
fy = np.array([])

for i in range(n):
fx = np.append(fx, gx[np.round(yt[i]).astype(int)] [np.round(xt[i]).astype(int)] )
fy = np.append(fy, gy[np.round(yt[i]).astype(int)] [np.round(xt[i]).astype(int)] )

xn = np.dot(inv, xt + gamma * fx) # acton, ivins equation
yn = np.dot(inv, yt + gamma * fy)

xt = xn
yt = yn

snake_final = np.array([xt, yt]).T
```

Gambar 4.14: *source code* proses update iterasi kurva versi *integer* dengan pembulatan kurva hanya di dalam iterasi

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dari eksperimen, maka dapat ditarik kesimpulan sebagai berikut:

1. Kontribusi peneliti terdapat pada penemuan dibutuhkannya tahap interpolasi *preprocessing* sebelum *active contour* dijalankan.

2. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* yang kurva akhirnya berhasil menutupi luka berjumlah 12 data (dari 71 data) sedangkan untuk yang versi interpolasi berjumlah 44 data (dari 71 data). Hal ini menunjukkan bahwa data yang berhasil dideteksi menggunakan *snake* interpolasi lebih banyak dibandingkan versi *integer*.

3. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* untuk semua kategori menghasilkan 12 data (dari 71 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 77.18%.

4. Hasil dari deteksi keliling luka menggunakan *snake* versi *integer* untuk semua kategori menghasilkan 44 data (dari 71 data) yang kurva akhirnya berhasil menutupi luka dengan nilai akurasi rata-rata 86.1%.

5.2 Saran

Penemuan tahap interpolasi sebelum *snake* dijalankan ini membuka peluang dilakukannya penelitian lanjutan untuk meningkatkan kinerja pada *snake* dengan

mengganti atau memodifikasi metode *snake* interpolasi dengan metode *preprocessing* citra lainnya demi meningkatkan hasil akurasi dari *snake*.

DAFTAR PUSTAKA

scikit-image active contour model. https://github.com/scikit-image/scikit-image/blob/v0.19.2/skimage/segmentation/active_contour_model.py. Accessed: 2022-08-05.

scikit-image rgb to grayscale. http:https://scikit-image.org/docs/0.19.x/auto_examples/color_exposure/plot_rgb_to_gray.html. Accessed: 2022-08-05.

scipy bivariate spline approximation over a rectangular mesh. https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RectBivariateSpline.html. Accessed: 2022-08-05.

Abdullah, M. A., Dlay, S. S., Woo, W. L., and Chambers, J. A. (2016). Robust iris segmentation method based on a new active contour force with a noncircular normalization. *IEEE transactions on systems, man, and cybernetics: Systems*, 47(12):3128–3141.

Acton, S. and Ray, N. (2007). Biomedical image analysis: Segmentation (synthesis lectures on image, video, & multimedia processing). *Morgan & Claypool Publishers*.

Amirullah, S. (2015). Metode penelitian manajemen. *Malang: Bayumedia Publishing Malang*.

Benbow, M. (2016). Best practice in wound assessment. *Nursing standard*, 30(27).

Brown, M. S., Ashley, B., and Koh, A. (2018). Wearable technology for chronic wound monitoring: current dressings, advancements, and future prospects. *Frontiers in bioengineering and biotechnology*, 6:47.

Budman, J., Keenahan, K., Acharya, S., and Brat, G. A. (2015). Design of a smartphone application for automated wound measurements for home care. *Iproceedings*, 1(1):e16.

Carrión, H., Jafari, M., Bagood, M. D., Yang, H.-y., Isseroff, R. R., and Gomez, M. (2022). Automatic wound detection and size estimation using deep learning algorithms. *PLoS computational biology*, 18(3):e1009852.

Constantia, E., Hidayat, B., et al. (2019). Estimasi bobot sapi berdasarkan registrasi citra digital dengan metode geometric active contour dan klasifikasi decision tree. *eProceedings of Engineering*, 6(1).

Fard, A. S., Esmaelzadeh, M., and Larijani, B. (2007). Assessment and treatment of

diabetic foot ulcer. *International journal of clinical practice*, 61(11):1931–1938.

Fraenkel, J. R., Wallen, N. E., and Hyun, H. H. (2012). *How to design and evaluate research in education*, volume 7. McGraw-hill New York.

Gay, L. R. and Diehl, P. (1992). *Research methods for business and management*. Macmillan Coll Division.

Gholami, P., Ahmadi-pajouh, M. A., Abolftahi, N., Hamarneh, G., and Kayvanrad, M. (2017). Segmentation and measurement of chronic wounds for bioprinting. *IEEE journal of biomedical and health informatics*, 22(4):1269–1277.

Gonzalez, R. C. and Woods, R. E. (2002). Digital image processing (preview).

Guo, M., Wang, Z., Ma, Y., and Xie, W. (2013). Review of parametric active contour models in image processing. *Journal of Convergence Information Technology*, 8(11):248.

Hemalatha, R., Thamizhvani, T., Dhivya, A. J. A., Joseph, J. E., Babu, B., and Chandrasekaran, R. (2018). Active contour based segmentation techniques for medical image analysis. *Medical and Biological Image Analysis*, 4(17):2.

HSE (2007). *Wound Management Guidelines*. USA.

Ickhsan, M. (2020). Implementasi metode segmentasi active contour untuk memperjelas tepi pada citra penyakit paru–paru. *Pelita Informatika: Informasi dan Informatika*, 8(3):357–360.

Ivins, J. and Porrill, J. (1995). Everything you always wanted to know about snakes (but were afraid to ask). *Artificial Intelligence*, 2000.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331.

Kesehatan, K. (2018). *Hasil utama riset kesehatan dasar (RISKESDAS) 2018*. Jakarta.

Kumar, R. V., Raju, K. P., Kumar, L. R., and Kumar, M. (2016). Gray level to rgb using ycbcr color space technique. *Int. J. Comput. Appl*, 147:25–28.

Landén, N. X., Li, D., and Ståhle, M. (2016). Transition from inflammation to proliferation: a critical step during wound healing. *Cellular and Molecular Life Sciences*, 73(20):3861–3885.

Monuteaux, M. C., Fleegler, E. W., and Lee, L. K. (2017). A cross-sectional study of emergency care utilization and associated costs of violent-related (assault) injuries

in the united states. *Journal of trauma and acute care surgery*, 83(5S):S240–S245.

Page, J. D. (2011). Algorithm to draw circles and ellipses. `https://www.mathopenref.com/coordcirclealgorithm.html`. Accessed: 2022-08-05.

Permata, E. (2015). Penggunaan metode active contour untuk segmentasi parasit malaria plasmodium falciparum. *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, 6(1):163–174.

Poon, T. W. K. and Friesen, M. R. (2015a). Algorithms for size and color detection of smartphone images of chronic wounds for healthcare applications. *IEEE Access*, 3:1799–1808.

Poon, T. W. K. and Friesen, M. R. (2015b). Algorithms for size and color detection of smartphone images of chronic wounds for healthcare applications. *IEEE Access*, 3:1799–1808.

Ratna Aryani, Muhammad Yusro, M. E. S. I. F. (2018). *BUKU PANDUAN : RANCANG BANGUN APLIKASI MOBILE ANDROID SEBAGAI ALAT DETEKSI WARNA DASAR LUKA DALAM MEMBANTU PROSES PENGKAJIAN LUKA KRONIS DENGAN NEKROSIS*. CRC Press.

Simon, P. E. (2018). Skin wound healing: Overview, hemostasis, inflammatory phase. https://emedicine.medscape.com/article/884594-overview.

Tyagi, V. (2018). *Understanding digital image processing*. CRC Press.

Van Poucke, S., Vander Haeghen, Y., Vissers, K., Meert, T., and Jorens, P. (2010). Automatic colorimetric calibration of human wounds. *BMC medical imaging*, 10(1):7.

Wang, L., Pedersen, P. C., Agu, E., Strong, D. M., and Tulu, B. (2016). Area determination of diabetic foot ulcer images using a cascaded two-stage svm-based classification. *IEEE Transactions on Biomedical Engineering*, 64(9):2098–2109.

Wang, L., Pedersen, P. C., Strong, D. M., Tulu, B., Agu, E., and Ignotz, R. (2014). Smartphone-based wound assessment system for patients with diabetes. *IEEE Transactions on Biomedical Engineering*, 62(2):477–488.

White, P. J., Podaima, B. W., and Friesen, M. R. (2014). Algorithms for smartphone and tablet image analysis for healthcare applications. *IEEE Access*, 2:831–840.

Wild, S., Roglic, G., Green, A., Sicree, R., and King, H. (2004). Global prevalence of diabetes: estimates for the year 2000 and projections for 2030. *Diabetes care*,

27(5):1047–1053.

William H. Press, Saul A. Teukolsky, W. T. V. and Flannery, B. P. *Numerical Recipes in C : The Art of Scientific Computing*.

Xu, C. and Prince, J. L. (1998). Snakes, shapes, and gradient vector flow. *IEEE Transactions on image processing*, 7(3):359–369.

Zhao, R., Liang, H., Clarke, E., Jackson, C., and Xue, M. (2016). Inflammation in chronic wounds. *International journal of molecular sciences*, 17(12):2085.

LAMPIRAN A

Source code program

1.1 main_integer.py

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
from skimage.filters import gaussian, sobel
from skimage.io import imread

my_dpi = 96 # https://www.infobyip.com/detectmonitordpi.php

# input data
kategori = "luka_merah"
path = "dataset_3/"+kategori+"/ready/"
path_skripsi = "D:/RL/skripsweet/dataset/dataset_3/"+ kategori +"/ready/"
img_name = "42"
extension = ".jpg"
img = imread(path + img_name + extension)
im_gray = rgb2gray(img)
im_gt = imread(path + img_name + "_g" + extension)


# parameter set
cr=160
cc=180
rad=90
sigma=3.5
sample=100
alpha = 1
beta = 10
gamma = 2 # time step
max_iter=100

# snake init (circle)
theta = np.linspace(0, 2*np.pi, sample)
r = cr + rad*np.sin(theta)
c = cc + rad*np.cos(theta)
snake_init = np.array([r, c]).T

snake_xy = snake_init[:, ::-1]
x = snake_xy[:, 0]
y = snake_xy[:, 1]
n = len(x)

fig0= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax0 = fig0.add_axes([0, 0, 1, 1])
ax0.imshow(im_gray, cmap=plt.cm.gray)
```

```
ax0.plot(snake_xy[:, 0], snake_xy[:, 1], '-r', lw=2)
ax0.set_xticks([]), ax0.set_yticks([]) # hide axes
ax0.axis('off')
outname0 = path + img_name +"_integer_init"+ extension
outname0_skripsi = path_skripsi + img_name +"_integer_init"+ extension
plt.savefig(outname0, dpi=my_dpi)
plt.savefig(outname0_skripsi, dpi=my_dpi)
print(outname0 + "␣has␣been␣saved")
print(outname0_skripsi + "␣has␣been␣saved")
plt.close(fig0)


# energy external
ext = gaussian(im_gray, sigma)
ext = sobel(ext)
ext = -ext**2

# save energy external
fig1= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax1 = fig1.add_axes([0, 0, 1, 1])
ax1.imshow(ext, cmap=plt.cm.gray)
ax1.set_xticks([]), ax1.set_yticks([]) # hide axes
ax1.axis('off')
outname1 = path + img_name +"_integer_ext"+ extension
outname1_skripsi = path_skripsi + img_name +"_integer_ext"+ extension
plt.savefig(outname1, dpi=my_dpi)
plt.savefig(outname1_skripsi, dpi=my_dpi)
print(outname1 + "␣has␣been␣saved")
print(outname1_skripsi + "␣has␣been␣saved")
plt.close(fig1)


# energy internal
# matrix
a = beta
b = -(4*beta + alpha)
c = 6*beta + 2*alpha

eye_n = np.eye(n, dtype=float)

c_axis = c * eye_n
b_axis = b * ( np.roll(eye_n, -1, axis=0) + np.roll(eye_n, -1, axis=1) )
a_axis = a * ( np.roll(eye_n, -2, axis=0) + np.roll(eye_n, -2, axis=1) )

A = c_axis + b_axis + a_axis
inv = np.linalg.inv(eye_n + gamma * A) # acton, ivins matrix equation

# potential force
gy, gx = np.gradient(ext)

# deform snake
fig2= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax2 = fig2.add_axes([0, 0, 1, 1])
ax2.imshow(im_gray, cmap=plt.cm.gray)
```

```
xt = np.copy(x)
yt = np.copy(y)

for i in range(max_iter):
fx = np.array([])
fy = np.array([])

for i in range(n):
fx = np.append(fx, gx[np.round(yt[i]).astype(int)] [np.round(xt[i]).astype(int)] )
fy = np.append(fy, gy[np.round(yt[i]).astype(int)] [np.round(xt[i]).astype(int)] )

xn = np.dot(inv, xt + gamma * fx) # acton, ivins equation
yn = np.dot(inv, yt + gamma * fy)

xt = xn
yt = yn

# ax2.plot(xt, yt, '-g', lw=2)

snake_final = np.array([xt, yt]).T

ax2.plot(snake_xy[:, 0], snake_xy[:, 1], '-r', lw=2)
ax2.plot(snake_final[:, 0], snake_final[:, 1], '-b', lw=2)
ax2.set_xticks([]), ax2.set_yticks([]) # hide axes
ax2.axis('off')
outname2 = path + img_name +"_integer_result"+ extension
outname2_skripsi = path_skripsi + img_name +"_integer_result"+ extension
plt.savefig(outname2, dpi=my_dpi)
plt.savefig(outname2_skripsi, dpi=my_dpi)
print(outname2 + "␣has␣been␣saved")
print(outname2_skripsi + "␣has␣been␣saved")
# plt.show()
plt.close(fig2)

# save final contour region
fig3= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax3 = fig3.add_axes([0, 0, 1, 1])
ax3.imshow(np.ones(img.shape))
ax3.fill(snake_final[:, 0], snake_final[:, 1], color="black")
ax3.set_xticks([]), ax3.set_yticks([]) # hide axes
ax3.axis('off')
outname3 = path + img_name +"_integer_r"+ extension
outname3_skripsi = path_skripsi + img_name +"_integer_r"+ extension
plt.savefig(outname3, dpi=my_dpi)
plt.savefig(outname3_skripsi, dpi=my_dpi)
print(outname3 + "␣has␣been␣saved")
print(outname3_skripsi + "␣has␣been␣saved")
plt.close(fig3)


# save overlay groundtruth & final snake

fig4= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
```

```
ax4 = fig4.add_axes([0, 0, 1, 1])
ax4.imshow(im_gt, cmap=plt.cm.gray)
ax4.plot(snake_final[:, 0], snake_final[:, 1], '-b', lw=2)
ax4.set_xticks([]), ax4.set_yticks([]) # hide axes
ax4.axis('off')
outname4 = path + img_name +"_gt_r_integer"+ extension
outname4_skripsi = path_skripsi + img_name +"_gt_r_integer"+ extension
plt.savefig(outname4, dpi=my_dpi)
plt.savefig(outname4_skripsi, dpi=my_dpi)
print(outname4 + "␣has␣been␣saved")
print(outname4_skripsi + "␣has␣been␣saved")
plt.close(fig4)
```

1.2 main_interpolation.py

```
import numpy as np
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
from skimage.util import img_as_float
from skimage.filters import gaussian, sobel
from skimage.io import imread
from scipy.interpolate import RectBivariateSpline
# skimage version 0.19.3

my_dpi = 96 # https://www.infobyip.com/detectmonitordpi.php

# input data
path = "dataset_3/luka_hitam/ready/"
path_skripsi = "D:/RL/skripsweet/dataset/dataset_3/luka_hitam/ready/"
img_name = "37"
extension = ".jpg"
img = imread(path + img_name + extension)
im_gray = rgb2gray(img)
im_gt = imread(path + img_name + "_g" + extension)

# parameter set
cr=110
cc=125
rad=80
sigma=3.5
sample=400
alpha = 0.015
beta = 10
gamma = 0.001 # time step
max_num_iter=500


# snake init (circle)
theta = np.linspace(0, 2*np.pi, sample)
r = cr + rad*np.sin(theta)
c = cc + rad*np.cos(theta)
```

```
snake_init = np.array([r, c]).T

snake_xy = snake_init[:, ::-1]
x = snake_xy[:, 0].astype(float)
y = snake_xy[:, 1].astype(float)
n = len(x)

fig0= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax0 = fig0.add_axes([0, 0, 1, 1])
ax0.imshow(im_gray, cmap=plt.cm.gray)
ax0.plot(snake_xy[:, 0], snake_xy[:, 1], '-r', lw=2)
ax0.set_xticks([]), ax0.set_yticks([]) # hide axes
ax0.axis('off')
outname0 = path + img_name +"_interp_init"+ extension
outname0_skripsi = path_skripsi + img_name +"_interp_init"+ extension
plt.savefig(outname0, dpi=my_dpi)
plt.savefig(outname0_skripsi, dpi=my_dpi)
print(outname0 + "␣has␣been␣saved")
print(outname0_skripsi + "␣has␣been␣saved")
plt.close(fig0)


# energy external
ext = gaussian(im_gray, sigma)
ext = img_as_float(ext)
ext = ext.astype(float, copy=False)
ext = sobel(ext)

# save energy external
fig1= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax1 = fig1.add_axes([0, 0, 1, 1])
ax1.imshow(ext, cmap=plt.cm.gray)
ax1.set_xticks([]), ax1.set_yticks([]) # hide axes
ax1.axis('off')
outname1 = path + img_name +"_interp_ext"+ extension
outname1_skripsi = path_skripsi + img_name +"_interp_ext"+ extension
plt.savefig(outname1, dpi=my_dpi)
plt.savefig(outname1_skripsi, dpi=my_dpi)
print(outname1 + "␣has␣been␣saved")
print(outname1_skripsi + "␣has␣been␣saved")
plt.close(fig1)


# energy internal

a = beta
b = -(4*beta + alpha)
c = 6*beta + 2*alpha

eye_n = np.eye(n, dtype=float)
c_axis = c * eye_n
b_axis = b * ( np.roll(eye_n, -1, axis=0) + np.roll(eye_n, -1, axis=1) )
a_axis = a * ( np.roll(eye_n, -2, axis=0) + np.roll(eye_n, -2, axis=1) )
A = c_axis + b_axis + a_axis
```

```
# Only one inversion is needed for implicit spline energy minimization. skimage
inv = np.linalg.inv(A + gamma * eye_n)
inv = inv.astype(float, copy=False)


# potential force
gy, gx = np.gradient(ext)

# Interpolate for smoothness
intp1 = RectBivariateSpline(np.arange(gx.shape[1]),
np.arange(gx.shape[0]),
gx.T, kx=2, ky=2, s=0)

intp2 = RectBivariateSpline(np.arange(gy.shape[1]),
np.arange(gy.shape[0]),
gy.T, kx=2, ky=2, s=0)


# deform snake
max_px_move=1.0

fig2= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax2 = fig2.add_axes([0, 0, 1, 1])
ax2.imshow(im_gray, cmap=plt.cm.gray)

xt = np.copy(x)
yt = np.copy(y)

for i in range(max_num_iter):
# fx & fy
fx = intp1(xt, yt, dx=0, grid=False).astype(float, copy=False)
fy = intp2(xt, yt, dy=0, grid=False).astype(float, copy=False)

# skimage equation
xn = np.dot(inv, gamma * xt + fx)
yn = np.dot(inv, gamma * yt + fy)

# Movements are capped to max_px_move per iteration. skimage
dx = max_px_move * np.tanh(xn - xt)
dy = max_px_move * np.tanh(yn - yt)

xt += dx
yt += dy

# if i % 10 == 0:
#         snake_iter = np.array([xt, yt]).T
#         ax2.plot(snake_iter[:, 0], snake_iter[:, 1], '-g', lw=2)

snake_final = np.array([xt, yt]).T

ax2.plot(snake_xy[:, 0], snake_xy[:, 1], '-r', lw=2)
ax2.plot(snake_final[:, 0], snake_final[:, 1], '-b', lw=2)
ax2.set_xticks([]), ax2.set_yticks([]) # hide axes
```

```
ax2.axis('off')
outname2 = path + img_name +"_interp_result"+ extension
outname2_skripsi = path_skripsi + img_name +"_interp_result"+ extension
plt.savefig(outname2, dpi=my_dpi)
plt.savefig(outname2_skripsi, dpi=my_dpi)
print(outname2 + "␣has␣been␣saved")
print(outname2_skripsi + "␣has␣been␣saved")
# plt.show()
plt.close(fig2)

# save final contour region
fig3= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax3 = fig3.add_axes([0, 0, 1, 1])
ax3.imshow(np.ones(img.shape))
ax3.fill(snake_final[:, 0], snake_final[:, 1], color="black")
ax3.set_xticks([]), ax3.set_yticks([]) # hide axes
ax3.axis('off')
outname3 = path + img_name +"_interp_r"+ extension
outname3_skripsi = path_skripsi + img_name +"_interp_r"+ extension
plt.savefig(outname3, dpi=my_dpi)
plt.savefig(outname3_skripsi, dpi=my_dpi)
print(outname3 + "␣has␣been␣saved")
print(outname3_skripsi + "␣has␣been␣saved")
plt.close(fig3)

# save overlay ground truth & final snake

fig4= plt.figure(frameon=False, figsize=(img.shape[1]/my_dpi, img.shape[0]/my_dpi), dpi=my_dpi)
ax4 = fig4.add_axes([0, 0, 1, 1])
ax4.imshow(im_gt, cmap=plt.cm.gray)
ax4.plot(snake_final[:, 0], snake_final[:, 1], '-b', lw=2)
ax4.set_xticks([]), ax4.set_yticks([]) # hide axes
ax4.axis('off')
outname4 = path + img_name +"_gt_r"+ extension
outname4_skripsi = path_skripsi + img_name +"_gt_r"+ extension
plt.savefig(outname4, dpi=my_dpi)
plt.savefig(outname4_skripsi, dpi=my_dpi)
print(outname4 + "␣has␣been␣saved")
print(outname4_skripsi + "␣has␣been␣saved")
plt.close(fig4)
```

1.3 integer_validation.py

```
import numpy as np
from skimage.color import rgb2gray
from skimage.io import imread
import matplotlib.pyplot as plt


path = "dataset_3/luka_merah/ready/"
img_name = "36"
```

```
extension = ".jpg"

groundtruth = rgb2gray(imread(path + img_name + "_r" + extension))
interp_region = rgb2gray(imread(path + img_name + "_integer_r" + extension))

sum_gt = 0
sum_interp = 0

for x in groundtruth:
for y in x:
if y != 1.0:
sum_gt += 1

for p in interp_region:
for q in p:
if q != 1.0:
sum_interp += 1

print(sum_gt)
print(abs(sum_gt-sum_interp))
print(abs(sum_gt-sum_interp) / sum_gt * 100)
print(100 - abs(sum_gt-sum_interp) / sum_gt * 100)
```

1.4 interp_validation.py

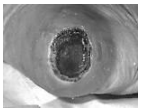
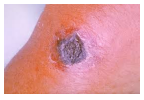
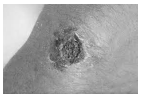
```
import numpy as np
from skimage.color import rgb2gray
from skimage.io import imread
import matplotlib.pyplot as plt


path = "dataset_3/luka_merah/ready/"
img_name = "44"
extension = ".jpg"

groundtruth = rgb2gray(imread(path + img_name + "_r" + extension))
interp_region = rgb2gray(imread(path + img_name + "_interp_r" + extension))

sum_gt = 0
sum_interp = 0

for x in groundtruth:
for y in x:
if y != 1.0:
sum_gt += 1

for p in interp_region:
for q in p:
if q != 1.0:
sum_interp += 1
```

```
print(sum_gt)
print(abs(sum_gt-sum_interp))
print(abs(sum_gt-sum_interp) / sum_gt * 100)
print(100 - abs(sum_gt-sum_interp) / sum_gt * 100)
```

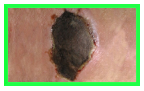
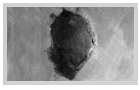
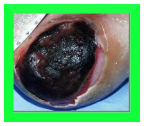
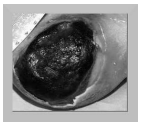
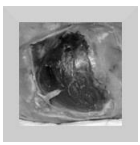
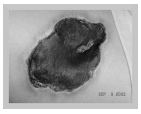
LAMPIRAN B

Tabel pengolahan data citra input

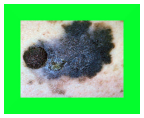
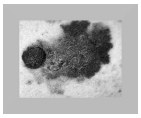
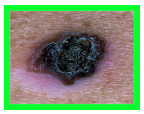
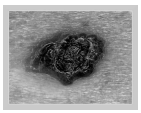
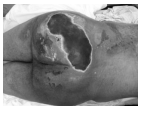
Tabel 2.1: Visualisasi hasil pengolahan data citra input luka hitam

No	File	Citra	*grayscale*	resolusi
1	luka_hitam/2.jpg			375x292
2	luka_hitam/4.jpg			295x292
3	luka_hitam/5.jpg			512x283
4	luka_hitam/6.jpg			350x263
5	luka_hitam/7.jpg			150x115
6	luka_hitam/8.jpg			350x263
7	luka_hitam/14.jpg			275x183

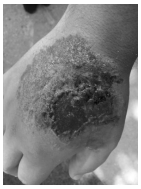
Tabel 2.2: Visualisasi hasil pengolahan data citra input luka hitam - lanjutan

No	File	Citra	*grayscale*	resolusi
8	luka_hitam/15.jpg			512x308
9	luka_hitam/17.jpg			283x247
10	luka_hitam/18.jpg			168x168
11	luka_hitam/19.jpg			512x374
12	luka_hitam/20.jpg			512x397
13	luka_hitam/22.jpg			390x276
14	luka_hitam/26.jpg			512x395

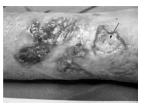
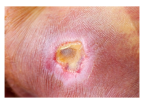
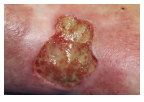
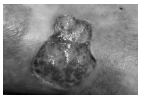
Tabel 2.3: Visualisasi hasil pengolahan data citra input luka hitam - lanjutan

No	File	Citra	*grayscale*	resolusi
15	luka_hitam/27.jpg			512x415
16	luka_hitam/28.jpg			234x216
17	luka_hitam/29.jpg			512x395
18	luka_hitam/33.jpg			375x250
19	luka_hitam/37.jpg			350x263
20	luka_hitam/40.jpg			262x192
21	luka_hitam/41.jpg			361x270

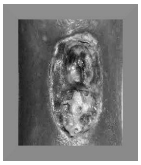
Tabel 2.4: Visualisasi hasil pengolahan data citra input luka hitam - lanjutan

No	**File**	**Citra**	***grayscale***	**resolusi**
22	luka_hitam/16.jpg			512x418
23	luka_hitam/31.jpg			383x512
24	luka_hitam/39.jpg			512x450

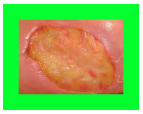
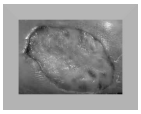
Tabel 2.5: Visualisasi hasil pengolahan data citra input luka kuning

No	File	Citra	*grayscale*	resolusi
1	luka_kuning/13.jpg			266x189
2	luka_kuning/17.jpg			259x194
3	luka_kuning/18.jpg			265x190
4	luka_kuning/19.jpg			120x151
5	luka_kuning/21.jpg			357x242
6	luka_kuning/23.jpg			500x333
7	luka_kuning/25.jpg			288x216

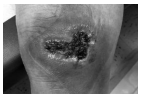
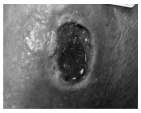
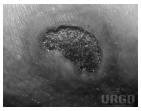
Tabel 2.6: Visualisasi hasil pengolahan data citra input luka kuning - lanjutan

No	File	Citra	*grayscale*	resolusi
8	luka_kuning/34.jpg			500x385
9	luka_kuning/35.jpg			242x208
10	luka_kuning/38.jpg			512x367
11	luka_kuning/42.jpg			512x384
12	luka_kuning/3.jpg			220x157
13	luka_kuning/12.jpg			500x333
14	luka_kuning/10.jpg			276x183

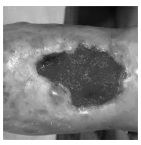
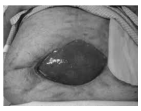
Tabel 2.7: Visualisasi hasil pengolahan data citra input luka kuning - lanjutan

No	**File**	**Citra**	***grayscale***	**resolusi**
15	luka_kuning/16.jpg			346x270

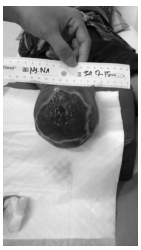
Tabel 2.8: Visualisasi hasil pengolahan data citra input luka merah

No	File	Citra	*grayscale*	resolusi
1	luka_merah/16.jpg			309x231
2	luka_merah/17.jpg			328x218
3	luka_merah/22.jpg			512x396
4	luka_merah/24.jpg			302x308
5	luka_merah/25.jpg			512x507
6	luka_merah/30.jpg			260x194
7	luka_merah/32.jpg			236x178

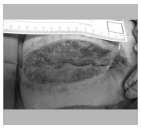
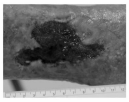
Tabel 2.9: Visualisasi hasil pengolahan data citra input luka merah - lanjutan

No	File	Citra	*grayscale*	resolusi
8	luka_merah/33.jpg			226x223
9	luka_merah/37.jpg			260x194
10	luka_merah/39.jpg			185x139
11	luka_merah/42.jpg			350x354
12	luka_merah/44.jpg			512x363
13	luka_merah/2.jpg			288x512
14	luka_merah/3.jpg			384x512

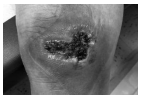
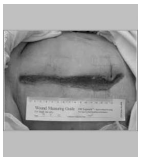
Tabel 2.10: Visualisasi hasil pengolahan data citra input luka merah - lanjutan

No	File	Citra	*grayscale*	resolusi
15	luka_merah/4.jpg			384x512
16	luka_merah/6.jpg			512x384
17	luka_merah/7.jpg			512x341
18	luka_merah/8.jpg			512x384
19	luka_merah/9.jpg			288x512
20	luka_merah/10.jpg			512x384
21	luka_merah/11.jpg			512x384

Tabel 2.11: Visualisasi hasil pengolahan data citra input luka merah - lanjutan

No	File	Citra	*grayscale*	resolusi
22	luka_merah/12.jpg			512x384
23	luka_merah/14.jpg			512x384
24	luka_merah/18.jpg			512x382
25	luka_merah/19.jpg			512x458
26	luka_merah/20.jpg			408x360
27	luka_merah/23.jpg			512x384
28	luka_merah/26.jpg			512x464

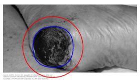
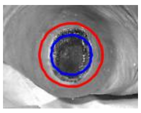
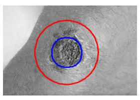
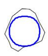
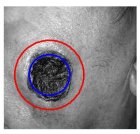
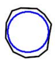
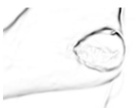
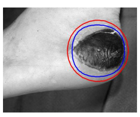
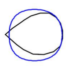
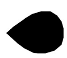
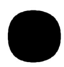
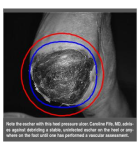
Tabel 2.12: Visualisasi hasil pengolahan data citra input luka merah - lanjutan

No	File	Citra	*grayscale*	resolusi
29	luka_merah/29.jpg			328x218
29	luka_merah/35.jpg			261x295
30	luka_merah/36.jpg			295x194
31	luka_merah/38.jpg			512x384
32	luka_merah/20.jpg			408x360

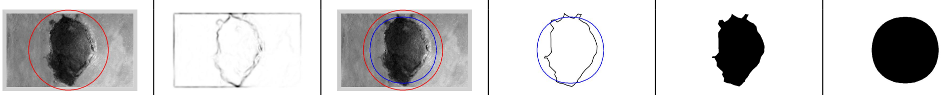
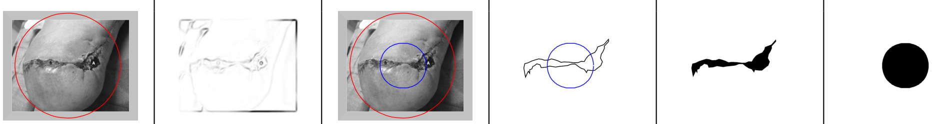
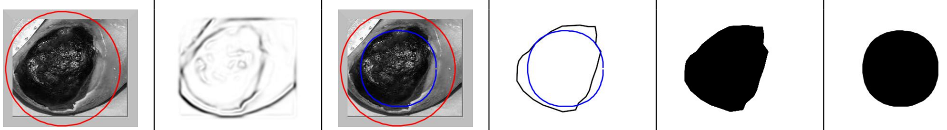
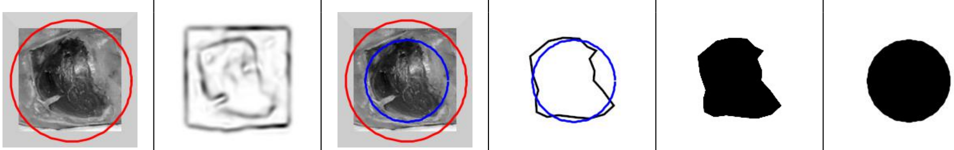
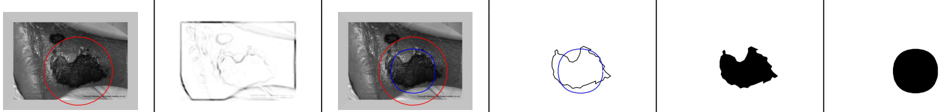
LAMPIRAN C

Tabel hasil deteksi keliling luka menggunakan *snake* versi *integer*

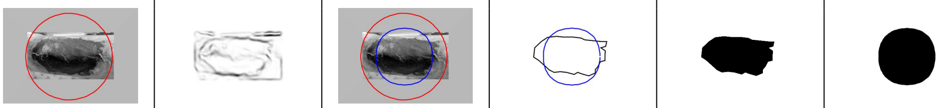
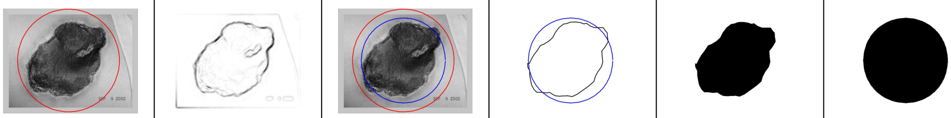
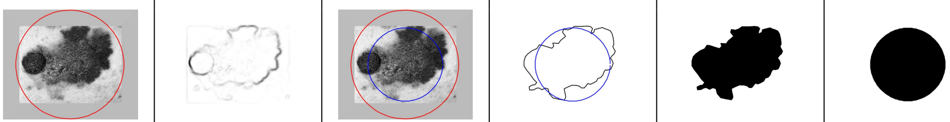
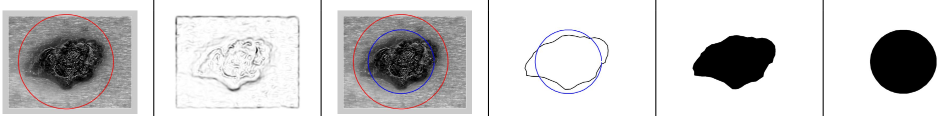
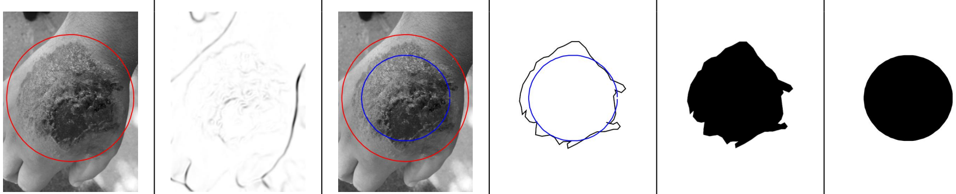
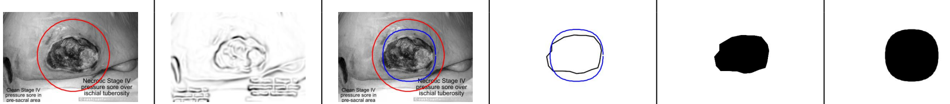
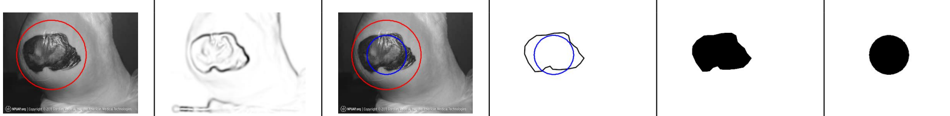
Tabel 3.1: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer*

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						gagal
						gagal

Tabel 3.2: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

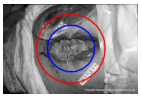
Tabel 3.3: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

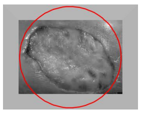
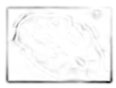
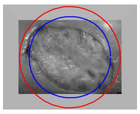
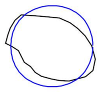
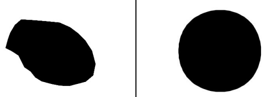
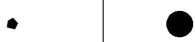
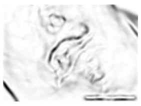
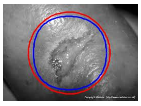
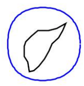
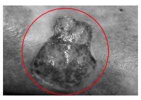
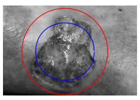
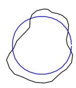
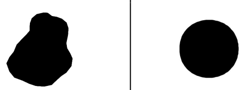
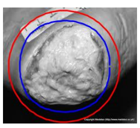
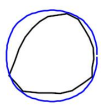
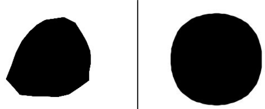
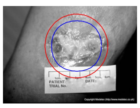
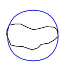
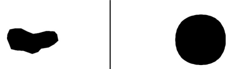
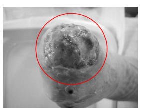
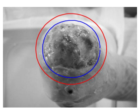
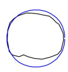
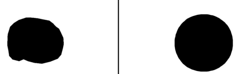
Tabel 3.4: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						gagal
						gagal

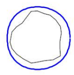
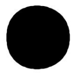
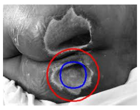
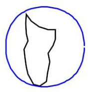
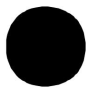
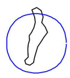
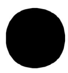
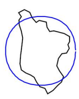
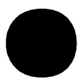
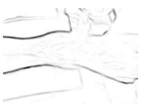
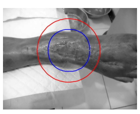
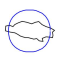
Tabel 3.5: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal

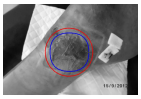
Tabel 3.6: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

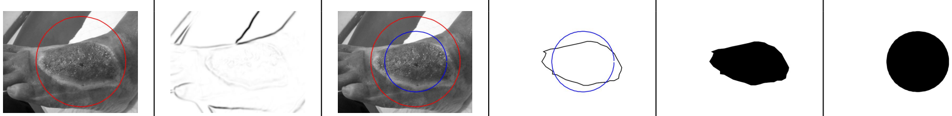
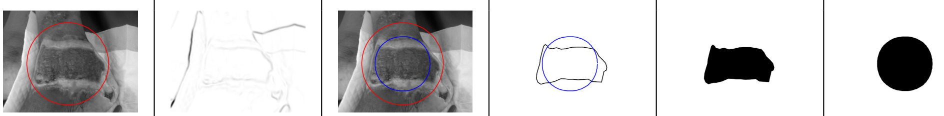
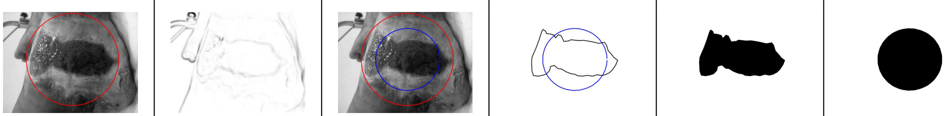
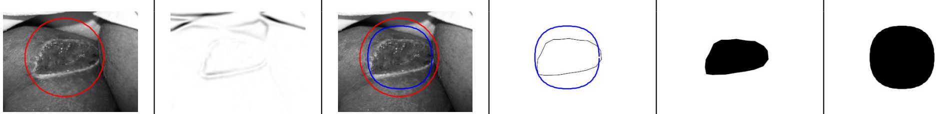
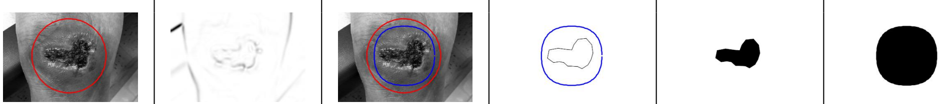
Tabel 3.7: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						gagal
						gagal
						gagal
						gagal

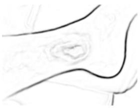
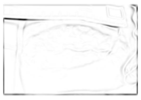
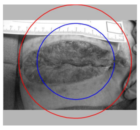
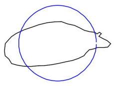
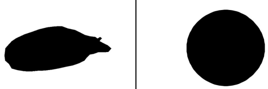
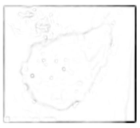
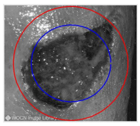
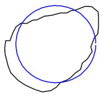
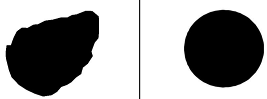
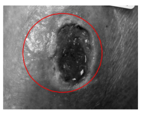
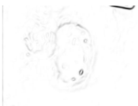
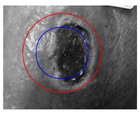
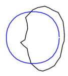
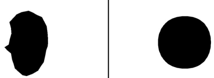
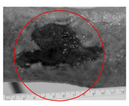
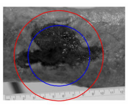
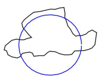
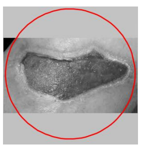
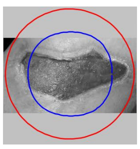
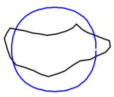
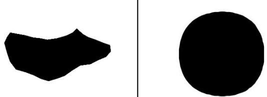
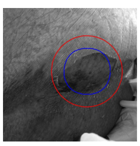
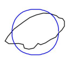
Tabel 3.8: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal

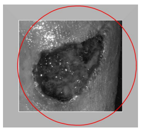
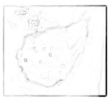
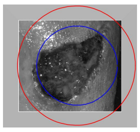
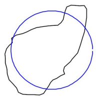
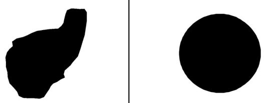
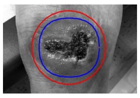
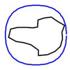
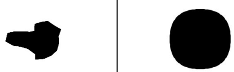
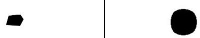
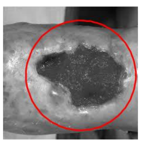
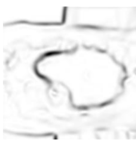
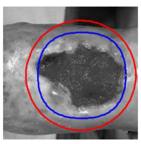
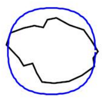
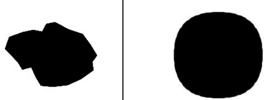
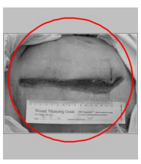
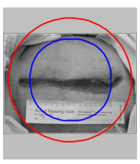
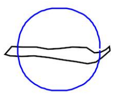
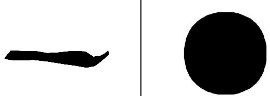
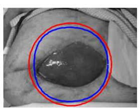
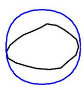
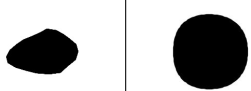
Tabel 3.9: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

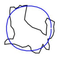
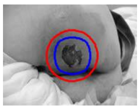
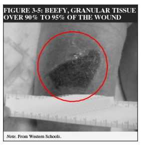
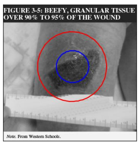
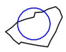
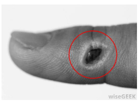
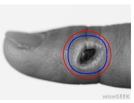
Tabel 3.10: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

Tabel 3.11: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

Tabel 3.12: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *integer* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal

luka hitam	integer		hasil deteksi berhasil menutupi luka							
file	cc	cr	r	sigma	sample	alpha	beta	time_step	max_iter	akurasi (%)
5	160	190	115	3.5	100	1	10	1	100	93.6
7	55	75	35	3.5	50	1	10	1	25	78.1
8	153	255	45	3.5	50	1	10	2	100	98.4
14	95	130	65	3.5	50	1	10	2	100	72.2
28	120	80	60	3.5	50	1	10	6	100	78.7
								rata-rata	=	84.2
populasi citra luka kategori luka_hitam						=	24			
sampel citra yang berhasil deteksi						=	5			
sampel citra yang gagal deteksi						=	19			

Gambar 3.1: parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka hitam

luka kuning		integer		hasil deteksi berhasil menutupi luka						
file	cc	cr	r	sigma	sample	alpha	beta	time_step	max_iter	akurasi (%)
13	70	206	50	3.5	100	1	10	2	50	37.9
19	60	60	50	3.5	100	1	10	2	100	84.1
21	130	170	70	3.5	100	1	10	2	100	73.8
25	145	130	55	3.5	100	1	10	1	100	85.8
34	185	245	130	3.5	100	1	10	1	100	99.1
								rata-rata	=	76.14
populasi citra luka kategori luka_kuning						=	15			
sampel citra yang berhasil deteksi						=	5			
sampel citra yang gagal deteksi						=	10			

Gambar 3.2: parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka kuning

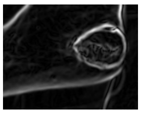
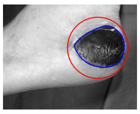
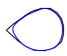
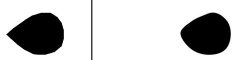
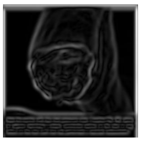
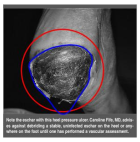
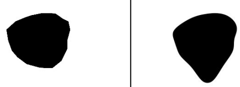
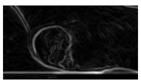
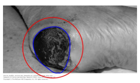
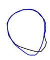
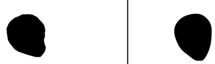
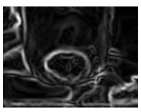
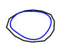
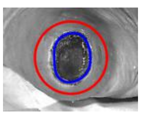
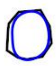
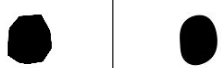
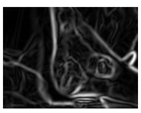
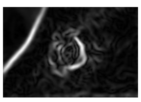
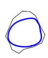
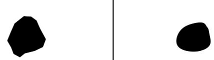
luka merah		integer		hasil deteksi berhasil menutupi luka						
file	cc	cr	r	sigma	sample	alpha	beta	time_step	max_iter	akurasi (%)
9	236	130	70	3.5	100	1	10	1	100	39.1
36	135	135	50	3.5	100	1	10	2	100	85.4
								rata-rata	=	62.25
populasi citra luka kategori luka_merah						=	32			
sampel citra yang berhasil deteksi						=	2			
sampel citra yang gagal deteksi						=	30			

Gambar 3.3: parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka merah

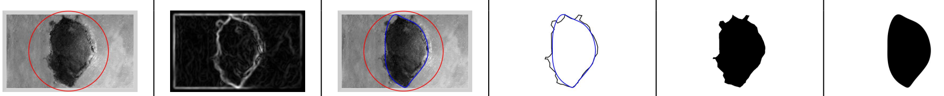
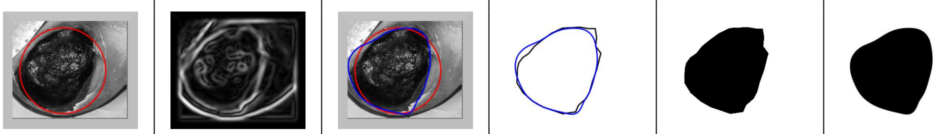
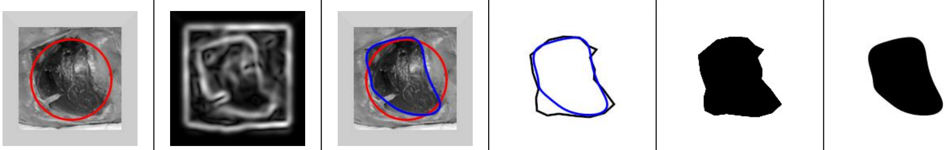
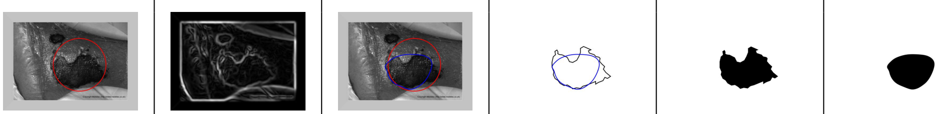
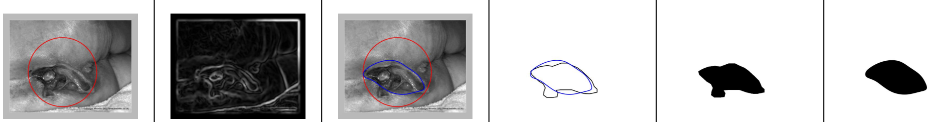
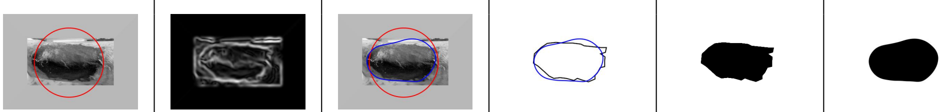
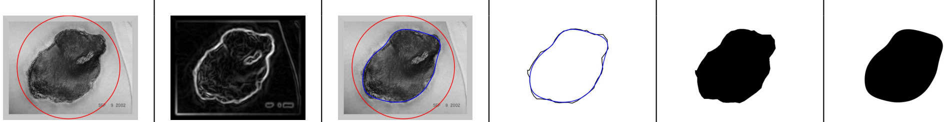
LAMPIRAN D

Tabel hasil deteksi keliling luka menggunakan *snake* versi *interpolasi*

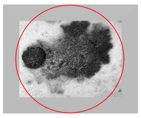
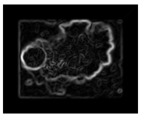
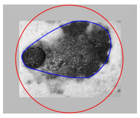
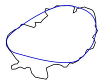
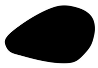
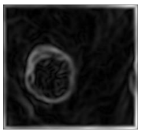
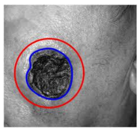
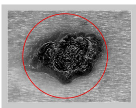
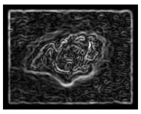
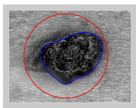
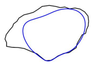
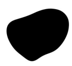
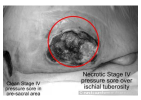
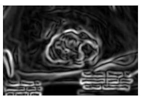
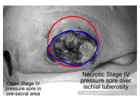
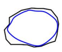
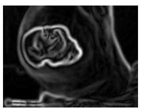
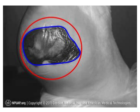
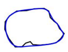
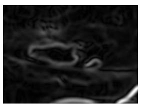
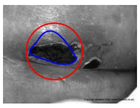
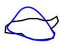
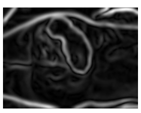
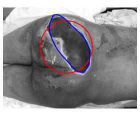
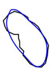
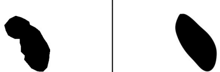
Tabel 4.1: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi*

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil

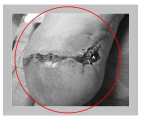
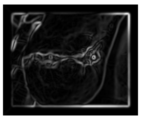
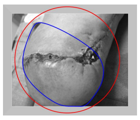
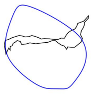
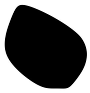
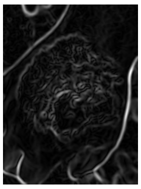
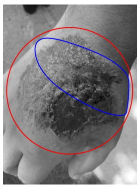
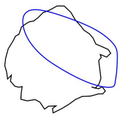
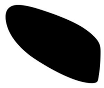
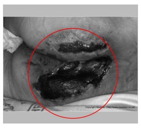
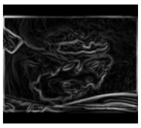
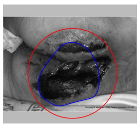
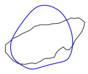
Tabel 4.2: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi*

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil

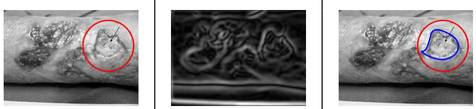
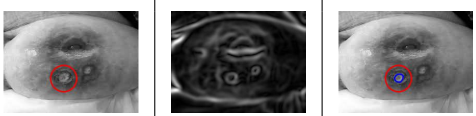
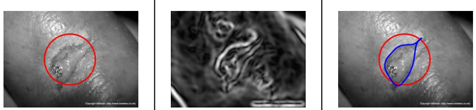
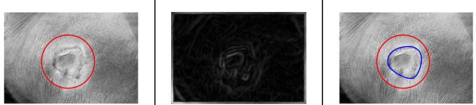
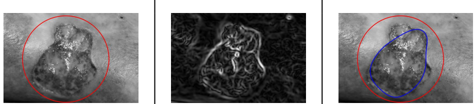
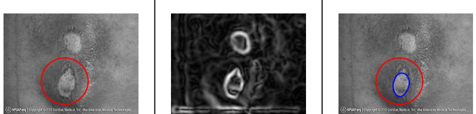
Tabel 4.3: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil

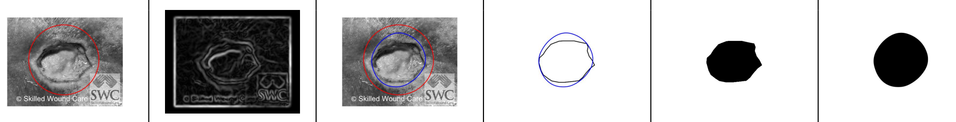
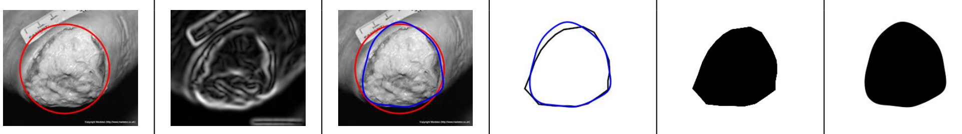
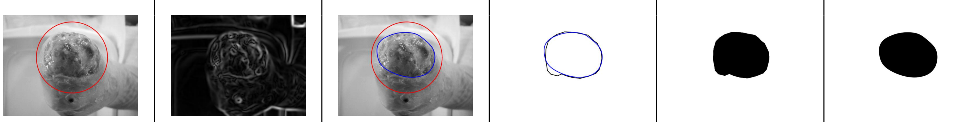
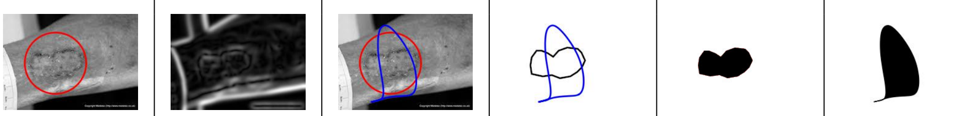
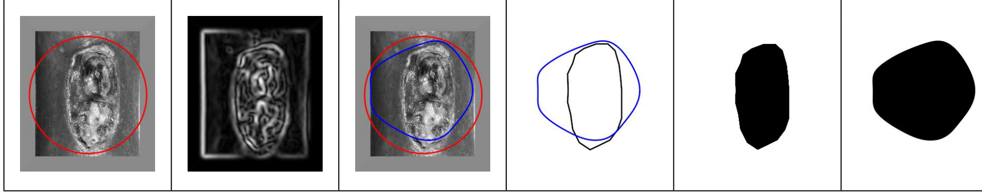
Tabel 4.4: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal

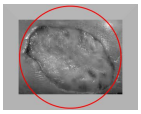
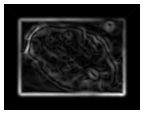
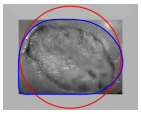
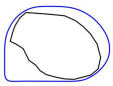
Tabel 4.5: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil

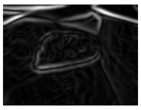
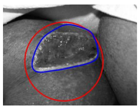
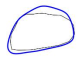
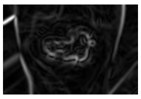
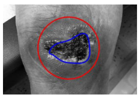
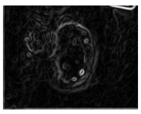
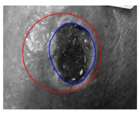
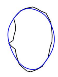
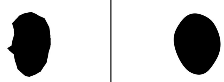
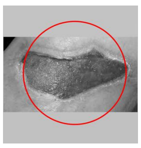
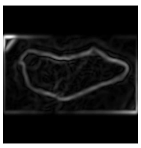
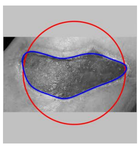
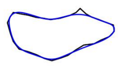
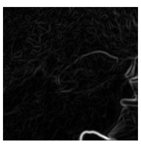
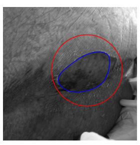
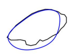
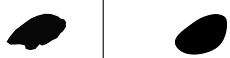
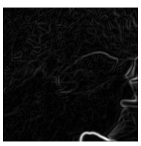
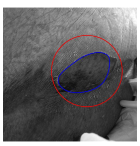
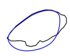
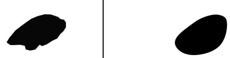
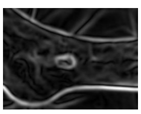
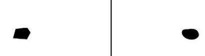
Tabel 4.6: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						gagal
						gagal
						gagal

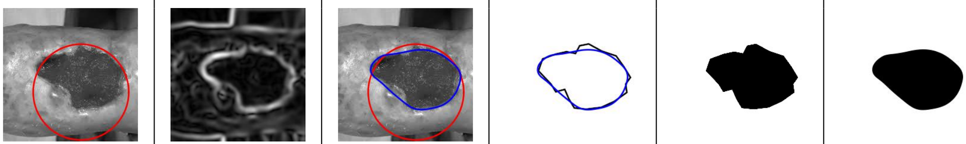
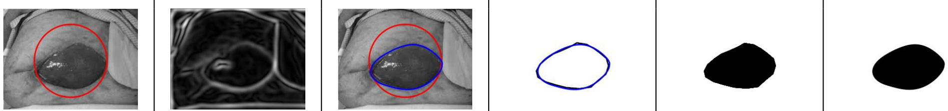
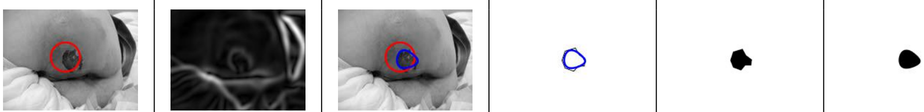
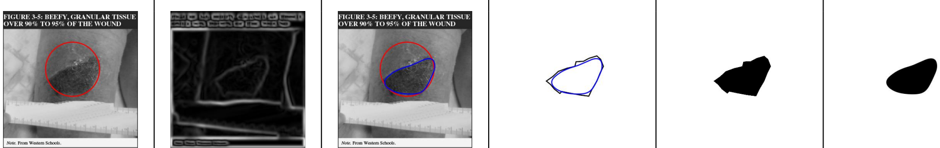
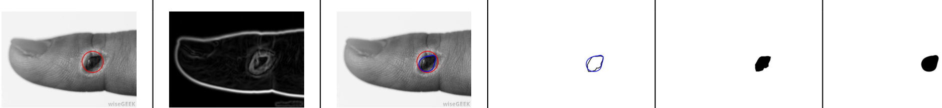
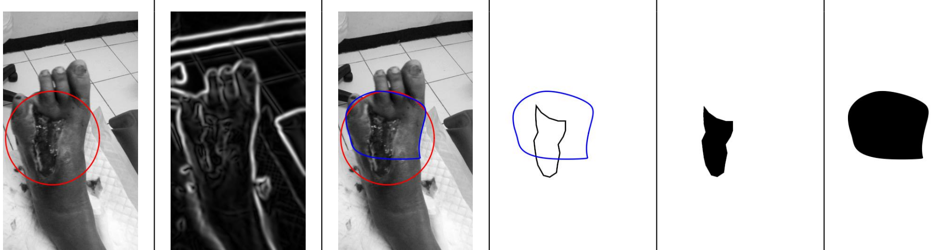
Tabel 4.7: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal

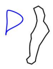
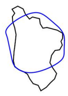
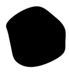
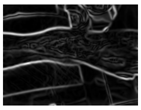
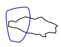
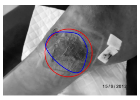
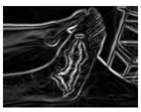
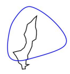
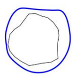
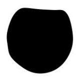
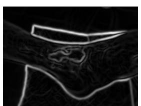
Tabel 4.8: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil

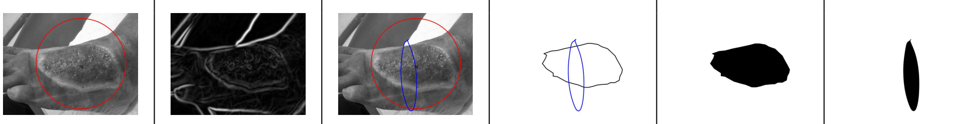
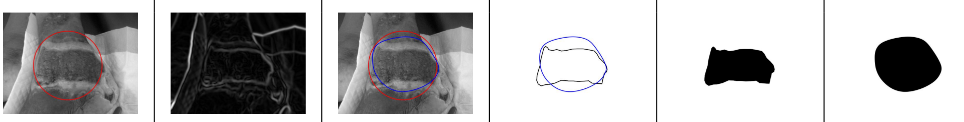
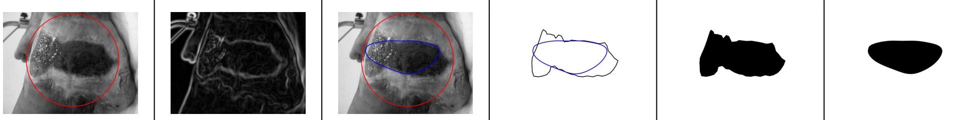
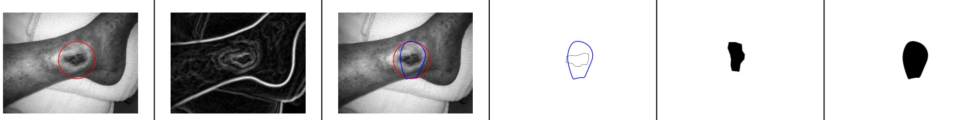
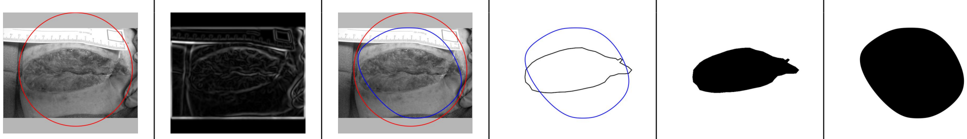
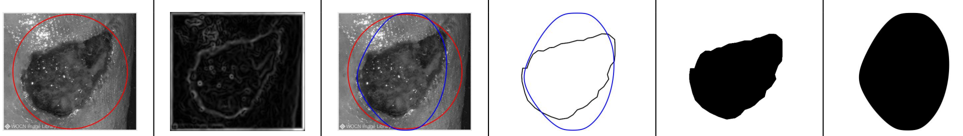
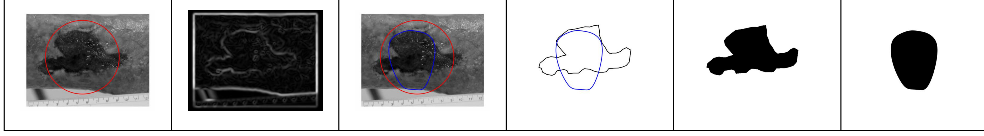
Tabel 4.9: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						berhasil
						gagal

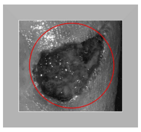
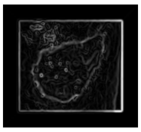
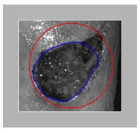
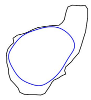
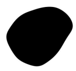
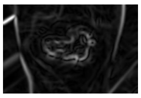
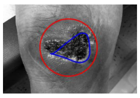
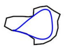
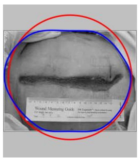
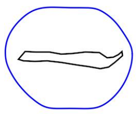
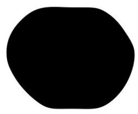
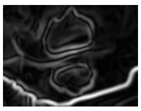
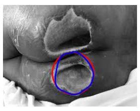
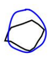
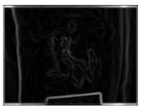
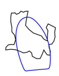
Tabel 4.10: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

Tabel 4.11: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal
						gagal

Tabel 4.12: Visualisasi hasil deteksi keliling luka menggunakan *snake* versi *interpolasi* - lanjutan

Inisialisasi	***Eksternal***	**kurva akhir**	**kurva akhir & *ground truth***	**area ground truth**	**area kurva**	**Kategori**
						gagal
						gagal
						gagal
						gagal
						gagal

luka hitam	interpolasi		hasil deteksi berhasil menutupi luka							
file	cc	cr	r	sigma	sample	alpha	beta	time_step	max_iter	akurasi (%)
2	120	265	85	3.5	400	0.015	10	0.001	800	94.5
4	45	130	90	3.5	400	0.015	10	0.001	500	85.2
5	160	190	115	3.5	400	0.015	10	0.001	500	96.6
6	155	165	70	3.5	400	0.015	10	0.001	500	79
7	55	75	40	3.5	400	0.015	10	0.001	500	87.2
8	153	255	35	3.5	400	0.015	10	0.001	100	80.3
14	95	130	65	3.5	400	0.015	10	0.001	500	76.2
15	153	250	151	3.5	400	0.015	10	0.001	500	96.8
17	124	125	90	3.5	400	0.015	10	0.001	500	96.7
18	85	85	50	3.5	400	0.015	10	0.001	500	87
19	200	290	100	3.5	400	0.015	10	0.001	500	82
20	220	225	130	3.5	400	0.015	10	0.001	500	98.6
22	140	190	100	3.5	400	0.015	10	0.001	500	85.7
26	195	248	193	3.5	400	0.015	10	0.001	500	97.8
27	207	253	205	3.5	400	0.015	10	0.001	500	87.3
28	120	80	60	3.5	400	0.015	10	0.001	500	94.3
29	195	240	160	3.5	400	0.015	10	0.001	500	74.7
33	100	195	70	3.5	400	0.015	10	0.001	500	81.6
37	110	125	80	3.5	400	0.015	10	0.001	500	94.2
40	90	100	55	3.5	400	0.015	10	0.001	500	70.8
41	105	170	70	5	400	0.015	10	0.001	500	93.8
										87.6333333
populasi citra luka kategori luka_hitam						=	24			
sampel citra yang berhasil deteksi						=	21			
sampel citra yang gagal deteksi						=	3			

Gambar 4.1: parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka hitam

luka kuning		interpolasi		hasil deteksi berhasil menutupi luka						
file	cc	cr	r	sigma	sample	alpha	beta	time_step	max_iter	akurasi (%)
13	70	206	50	3.5	400	0.015	10	0.001	500	71
17	128	115	25	3.5	400	0.015	10	0.001	500	77.8
18	95	130	50	3.5	400	0.015	10	0.001	500	98.5
19	60	60	50	3.5	200	0.015	10	0.001	40	85.3
21	130	170	70	3	200	0.015	10	0.001	500	80.3
23	170	230	160	3.5	400	0.015	10	0.001	100	85.2
25	145	130	50	3.5	200	0.015	10	0.001	500	55.4
34	185	245	130	3.5	400	0.015	10	0.001	500	71.2
35	105	110	80	3.5	400	0.015	10	0.001	500	92.5
38	150	280	100	3.5	400	0.015	10	0.001	500	57.5
42	160	260	135	3.5	400	0.015	10	0.001	500	96.7
								rata-rata	=	79.21818182
populasi citra luka kategori luka_kuning						=	15			
sampel citra yang berhasil deteksi						=	11			
sampel citra yang gagal deteksi						=	4			

Gambar 4.2: parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka kuning

luka merah		interpolasi		hasil deteksi berhasil menutupi luka						
file	cc	cr	r	sigma	sample	alpha	beta	time_step	max_iter	akurasi (%)
16	130	140	90	3.5	400	0.015	10	0.001	500	71.5
17	105	160	75	3.5	400	0.015	10	0.001	400	90.6
22	180	225	150	3.5	400	0.015	10	0.001	800	93.3
24	150	160	115	3.5	400	0.015	10	0.001	400	98.2
25	240	320	135	3.5	400	0.015	10	0.001	400	90.2
30	105	120	30	3.5	400	0.015	10	0.001	150	97.3
32	83	120	70	3.5	400	0.015	10	0.001	400	81.1
33	140	130	80	3.5	400	0.015	10	0.001	400	99.5
37	100	130	170	3.5	400	0.015	10	0.001	400	94.6
39	65	85	20	3.5	400	0.015	10	0.001	100	99.8
42	150	180	70	3.5	400	0.015	10	0.001	500	89.8
44	190	345	70	3.5	400	0.015	10	0.001	500	70.9
								rata-rata	=	89.73333333
populasi citra luka kategori luka_merah						=	32			
sampel citra yang berhasil deteksi						=	12			
sampel citra yang gagal deteksi						=	20			

Gambar 4.3: parameter dan akurasi dari hasil deteksi keliling luka menggunakan *snake* versi *integer* untuk kategori luka merah