

## Tutorial Singkat v.1.0

### RANSAC - Teknik optimisasi data berbasis *iterative model fitting*

Oleh: Sunu Wibirama, M.Eng<sup>1</sup>

Pada sebuah eksperimen yang melibatkan puluhan, ratusan atau bahkan ribuan data, kita seringkali dihadapkan pada sejumlah data yang memiliki relasi linier. Dari data-data tersebut, biasanya kita akan menjelaskan relasi linier antar data dengan menurunkan sebuah persamaan linier melalui teknik regresi atau *least-square*. Apabila data-data tersebut memiliki relasi linier yang cukup sempurna, model persamaan yang diturunkan pun akan mengandung kesalahan yang cukup kecil. Namun bila data-data tersebut memiliki relasi linier yang tidak terlalu baik (dengan berbagai macam *noise* atau *outliers*), model persamaan yang dihasilkan akan berubah dan bukan tidak mungkin memiliki tingkat akurasi yang cukup rendah. Ketidakakuratan data ini bisa muncul dari beberapa hal, antara lain:

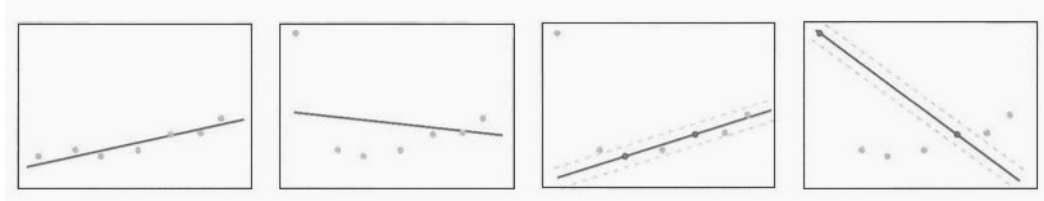
- Kesalahan pada proses pengukuran: pada kasus observasi yang melibatkan puluhan atau ratusan data, seringkali kita menghadapi beberapa titik data yang tidak sesuai dengan kondisi yang kita inginkan. Pada tutorial computer vision, hal ini banyak dijumpai saat kita mendeteksi *feature points* di sebuah citra.
- Kesalahan klasifikasi: seringkali kesalahan terletak pada proses klasifikasi titik-titik data (mis-identified). Apabila hal ini terjadi, kesalahan yang muncul pada model persamaan yang akan diturunkan dari data tersebut biasanya cukup besar

Menyikapi kondisi ketidakakuratan tersebut, diperlukan sebuah metode yang efisien untuk mendeteksi *outliers*, yakni titik-titik data yang memiliki jarak terbesar dari model persamaan yang akan diturunkan. Bila titik-titik data yang dianggap sebagai *outliers* sudah bisa ditemukan, *outliers* tersebut bisa disisihkan dan model persamaan tersebut bisa dihitung ulang dengan titik-titik data yang tersisa.

Pada teknik *least-square*, asumsi yang digunakan adalah semakin banyak data yang digunakan, persamaan linier yang dihasilkan semakin baik. Namun pada beberapa kondisi tertentu, asumsi yang digunakan adalah sebaliknya, yakni semakin sedikit data yang digunakan akan semakin baik untuk menghasilkan persamaan linier, sebab apabila data-data yang digunakan mengandung banyak *outliers*, maka model yang dihasilkan akan memiliki akurasi rendah, sebagaimana tergambar pada ilustrasi di bawah ini.

---

<sup>1</sup> Penulis adalah pengajar dan peneliti di Jurusan Teknik Elektro dan Teknologi Informasi, Fakultas Teknik, Universitas Gadjah Mada, Yogyakarta. Penulis bisa dihubungi melalui email: sunu[at]ugm.ac.id



**Gambar 1.** Beberapa hasil model fitting dengan menggunakan algoritma *least-square*. Noise sangat berpengaruh terhadap model yang dihasilkan algoritma *least-square*.

Untuk kasus *model fitting*, dua buah titik data sudah cukup untuk menghasilkan satu buah garis. Garis ini dapat digunakan untuk mendefinisikan model persamaan linier awal yang akan digunakan untuk menghasilkan persamaan linier terbaik dari titik-titik data yang kami miliki. Model awal ini bisa diuji dengan melakukan observasi, seberapa banyak titik-titik data yang dimiliki berada di “sekitar” atau “dekat” dengan model awal tersebut. Data-data inilah yang kemudian sebagai titik konsensus (*consensus points*) dan digunakan untuk menentukan model terbaik setelah melalui beberapa iterasi perhitungan. Teknik “memulai dari sedikit titik untuk menemukan model relasi linier dari data” inilah yang kemudian disebut sebagai algoritma **Random Sample Consensus** [1].

Secara garis besar, algoritma ini memiliki prosedur perhitungan sebagai berikut:

- a. Asumsikan titik data sejumlah  $n$ , yakni  $X = \{x_1, x_2, \dots, x_n\}$ , di mana sebuah model relasi linier dari data tersebut dibuat dengan menggunakan setidaknya  $m$  titik, dimana  $m \leq n$ . Untuk menghasilkan sebuah garis sempurna, setidaknya dibutuhkan 2 titik.
- b. Mulai *counter* iterasi  $k = 1$
- c. Pilih secara acak  $m$  buah data dari  $X$ , kemudian hitung sebuah model linier
- d. Untuk sebuah nilai toleransi  $\epsilon$  dari model yang dihasilkan, tentukan seberapa banyak elemen  $X$  yang berada dalam nilai toleransi tersebut. Jika jumlah dari elemen-elemen ini melebihi sebuah nilai ambang batas  $t$ , maka hitung kembali model menggunakan titik-titik konsensus, kemudian hentikan algoritma.
- e. Naikan nilai *counter* menjadi  $k = k+1$ . Jika  $k < K$ , untuk sebuah nilai  $K$  yang ditentukan di awal perhitungan, maka ulangi langkah *c*. Apabila terjadi selainnya, maka terima hasil perhitungan model linier dengan jumlah titik-titik konsensus terbesar.

Pada algoritma RANSAC rumus umum untuk menghitung jumlah iterasi yang diperlukan, yakni:

$$K = \frac{\log \zeta}{\log(1 - \xi^M)} \quad (1)$$

**Dengan:**

K = jumlah iterasi yang diperlukan untuk menemukan model linier terbaik

M = jumlah titik data yang diperlukan untuk menghitung model linier

Q = jumlah inliers atau data yang berada di sekamir model

N = jumlah data keseluruhan

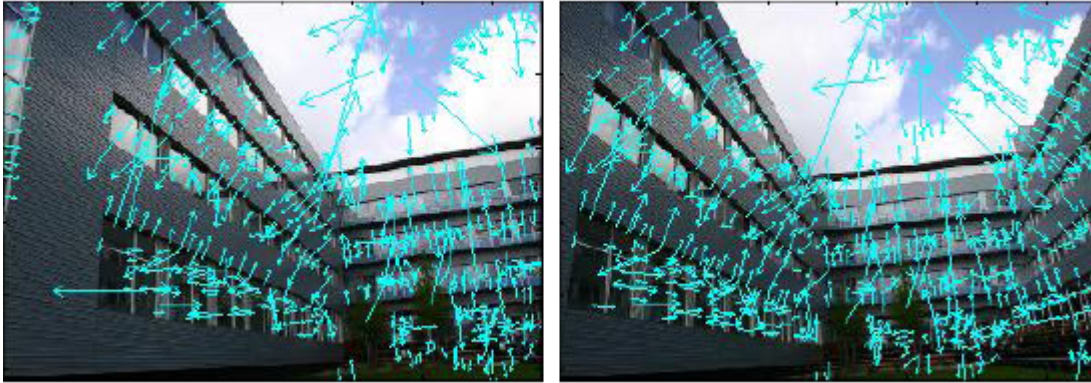
$\xi$  = dibaca *xi*, yakni rasion inliers terhadap seluruh data (Q/N)

$\zeta$  = dibaca *zeta*, yakni peluang RANSAC tidak menemukan solusi yang benar dan bisa diterima.

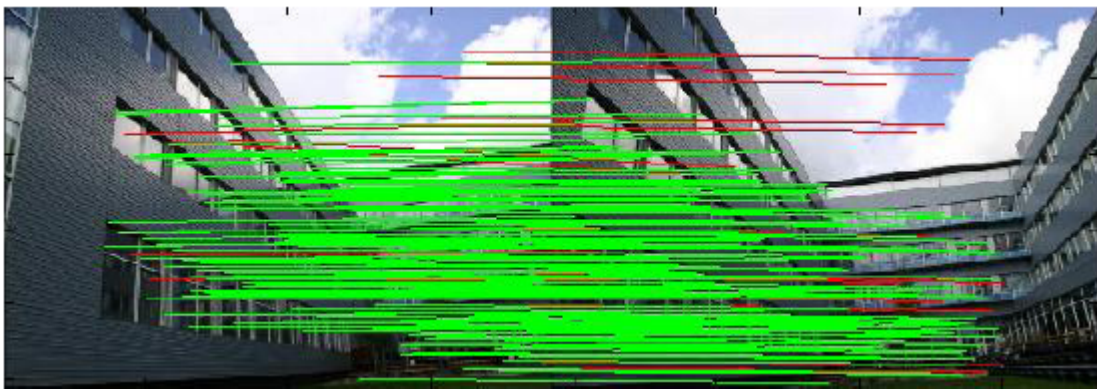
Gambar 2 sampai dengan Gambar 5 menunjukkan salah satu implementasi RANSAC untuk *panoramic stitching* (menggabungkan dua citra panorama yang diproduksi dengan mengambil gambar dari dua sudut pandang yang berbeda). Pada Gambar 2, dua buah citra gedung diambil dari sudut pandang yang berbeda. Dua citra ini nantinya akan menjadi masukan untuk algoritma panoramic stitching. Gambar 3 menunjukkan hasil deteksi fitur pada dua citra. Seleksi dengan RANSAC dilakukan untuk memisahkan *inliers* dari *outliers* (Gambar 4). *Panoramic stitching* dilakukan dengan menggunakan fitur yang dikategorikan sebagai *good matching* (Gambar 5).



**Gambar 2.** Dua citra masukan sebelum diproses



**Gambar 3.** Deteksi fitur pada dua buah citra



**Gambar 4.** Seleksi *inliers* dan *outliers* dengan RANSAC.

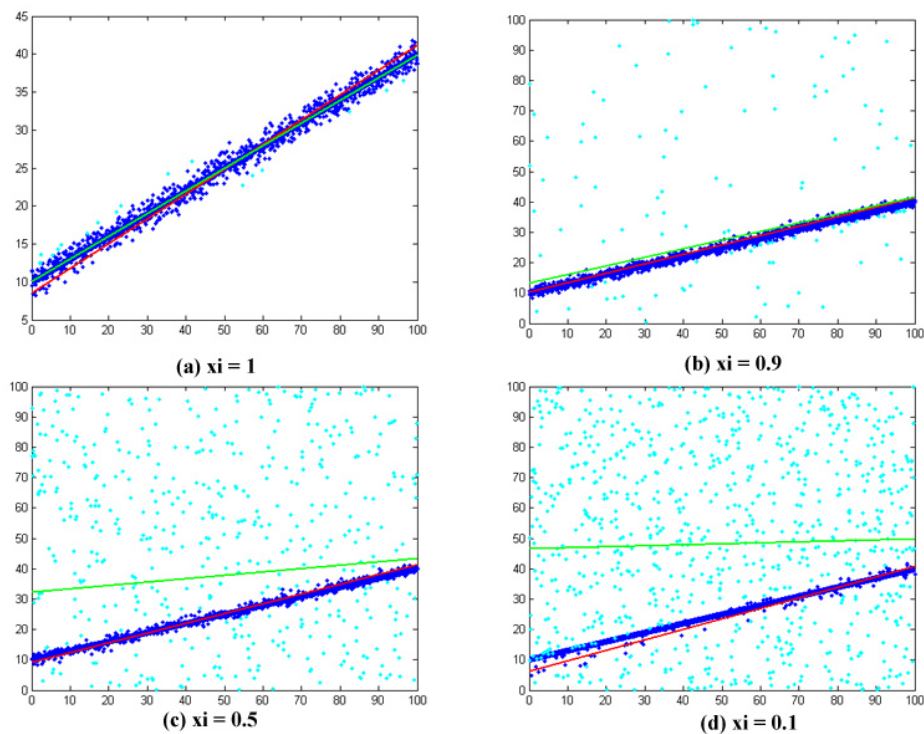
Garis hijau menunjukkan fitur diklasifikasikan sebagai *inliers* (*good matching*), sedangkan garis merah menunjukkan fitur yang diklasifikasikan sebagai *outliers* (*bad matching*).



**Gambar 5.** Hasil panoramic stitching

Pada tutorial ini, pengaruh kuantitas *outliers* terhadap jumlah iterasi RANSAC juga disimulasikan. Perangkat lunak MATLAB digunakan untuk melakukan simulasi. Adapun asumsi awal simulasi sebagai berikut:

- Jumlah data  $N = 1000$
- Model awal :  $y = 10 + 0.3x$
- Jumlah inliers:  $\xi N \rightarrow$  Jumlah outliers:  $(1 - \xi)N$



**Gambar 6.** Simulasi optimisasi pemodelan liner dengan RANSAC. Hasil perhitungan dengan menggunakan algoritma *least-square* adalah garis hijau. Garis merah adalah hasil perhitungan dengan algoritma RANSAC. Titik-titik biru adalah *inliers*, sedangkan titik biru muda adalah outliers atau *noise*.

**Tabel 1.** Hasil Percobaan Optimisasi Pemodelan Linier dengan RANSAC

Percobaan	Nilai $\xi$ (ratio <i>inliers</i> terhadap <i>N</i> )	Jumlah iterasi ( <i>K</i> )
1 (Gambar 17.a)	1	2
2 (Gambar 17.b)	0.9	3
3 (Gambar 17.c)	0.5	15
4 (Gambar 17.d)	0.1	148

Hasil simulasi ditunjukkan oleh Gambar 6. Hasil perhitungan dengan menggunakan algoritma *least-square* ditunjukkan dengan garis hijau. Garis merah adalah hasil perhitungan dengan algoritma RANSAC. Titik-titik biru adalah *inliers*, sedangkan titik biru muda adalah *outliers* atau *noise*.

Pada Gambar 6.a, nilai dari  $\xi$  adalah 1, yang berarti sama sekali tidak ada *outliers* dalam data tersebut. Hasil perhitungan menunjukkan, jumlah iterasi yang diperlukan sangat sedikit, yakni 2 iterasi untuk menghasilkan model persamaan linier yang sesuai dengan asumsi awal. Seiring dengan meningkatnya jumlah *outliers* (menurunnya nilai  $\xi$ ), jumlah iterasi yang diperlukan oleh RANSAC untuk menghasilkan model persamaan linier yang tepat semakin besar. Pada Gambar 6.d ditunjukkan, untuk jumlah *inliers* 100 titik ( $0.1 * 1000$ ) dan jumlah *outliers* 900 titik ( $0.9 * 1000$ ), RANSAC membutuhkan 148 iterasi untuk memperoleh model persamaan linier yang tepat. Dari hasil tutorial juga diperoleh, algoritma *least-square* mengalami *error* yang cukup besar apabila *outliers* lebih dominan daripada *inliers*.

Dengan demikian dapat kita tarik sebuah kesimpulan dari simulasi ini, bahwa semakin besar *outliers*, iterasi yang dibutuhkan akan semakin banyak (lihat Tabel 1). Jika diasumsikan jumlah *outliers* tidak diketahui, maka untuk mendapatkan model persamaan linier yang optimal perlu ada *iterasi* yang cukup.

Implementasi RANSAC dapat ditemukan dengan mudah di internet, dua diantaranya yang dapat dijadikan bahan belajar dan rujukan terdapat pada referensi [2, 3]. Dengan memanfaatkan RANSAC, algoritma *stitching* atau *matching* berbasis *feature points* menjadi lebih optimal.

## REFERENSI

- [1] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381-395, 1981.
- [2] P. Kovesi, "Matlab and Octave Functions for Computer Vision and Image Processing:", Available: <http://www.csse.uwa.edu.au/~pk/research/matlabfns/>
- [3] C. Kenney, M. Bober, and B.S. Manjunath, "RANSAC Toolbox", Available: <http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/RANSAC.shtml>