

SKRIPSI

«JUDUL BAHASA INDONESIA»



Prayogo Cendra

NPM: 2014730033

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN

«tahun»

UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»



Prayogo Cendra

NPM: 2014730033

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

LEMBAR PENGESAHAN

«JUDUL BAHASA INDONESIA»

Prayogo Cendra

NPM: 2014730033

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Claudio Franciscus, M.T.

«pembimbing pendamping/2»

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

«JUDUL BAHASA INDONESIA»

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Meterai Rp. 6000

Prayogo Cendra
NPM: 2014730033

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini...?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini ...»

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	2
2 LANDASAN TEORI	5
2.1 Linear Programming	5
2.1.1 Karakteristik	5
2.1.2 Daerah Solusi dan Solusi Optimal	6
2.1.3 Bentuk Standar	8
2.1.4 Variabel Basis dan Non Basis	9
2.1.5 Metode Simplex	9
2.2 Integer Programming	12
2.2.1 Branch and Bound	12
2.3 Template Skripsi FTIS UNPAR	13
2.3.1 Tabel	13
2.3.2 Kutipan	14
2.3.3 Gambar	14
3 ANALISIS	19
3.1 Pemodelan Masalah	19
3.1.1 Ruang	19
3.1.2 Kamera CCTV	19
3.1.3 Penempatan Kamera CCTV	20
3.1.4 Daerah Cakupan	20
3.2 Penyelesaian Masalah	23
3.2.1 Variabel Keputusan	23
3.2.2 Fungsi Tujuan	24
3.2.3 Batasan	24
3.2.4 Bentuk Linear Programming	25
3.3 Analisis Kebutuhan Prangkat Lunak	25
3.3.1 Diagram <i>Use Case</i>	25

3.3.2	Diagram Kelas	26
4	PERANCANGAN	27
4.1	Perancangan Antarmuka	27
4.2	Perancangan Kelas	28
4.2.1	Kelas <i>Angle</i>	30
4.2.2	Kelas <i>Point</i>	31
4.2.3	Kelas <i>CameraSpecification</i>	32
4.2.4	Kelas <i>CameraPlacement</i>	32
4.2.5	Kelas <i>Dimension</i>	33
4.2.6	Kelas <i>Cell</i>	33
4.2.7	Kelas <i>GridPoint</i>	34
4.2.8	Kelas <i>Room</i>	35
4.2.9	Kelas <i>MinimumCameraPlacementSolver</i>	36
4.2.10	Kelas <i>MinimumCameraPlacementSolverLPSolve</i>	37
	DAFTAR REFERENSI	39
	A KODE PROGRAM	41
	B HASIL EKSPERIMEN	43

DAFTAR GAMBAR

2.1	Daerah solusi	6
2.2	Daerah solusi yang dibatasi	7
2.3	Daerah solusi yang tidak dibatasi	7
2.4	Tidak ada daerah solusi	8
2.5	Gambar <i>Serpentes</i> dalam format png	16
2.6	Ular kecil	16
2.7	<i>Serpentes</i> betina	17
3.1	Pemodelan ruangan	19
3.2	Pemodelan kamera CCTV	20
3.3	Pemodelan penempatan kamera CCTV	20
3.4	Pemodelan ruangan dalam bentuk grid point	21
3.5	Daerah cakupan sebelum pemodelan grid point	22
3.6	Daerah cakupan sesudah pemodelan grid point	22
3.7	Daerah overlap dan out of bound	23
3.8	Diagram <i>use case</i>	25
3.9	Diagram kelas sederhana	26
4.1	Antarmuka penerima masukan	27
4.2	Antarmuka penempatan kamera CCTV	28
4.3	Diagram kelas lengkap	29
4.4	Diagram kelas <i>Angle</i>	30
4.5	Diagram kelas <i>Point</i>	31
4.6	Diagram kelas <i>CameraSpecification</i>	32
4.7	Diagram kelas <i>CameraPlacement</i>	32
4.8	Diagram kelas <i>Dimension</i>	33
4.9	Diagram kelas <i>Cell</i>	33
4.10	Diagram kelas <i>GridPoint</i>	34
4.11	Diagram kelas <i>Room</i>	35
4.12	Diagram kelas <i>MinimumCameraPlacementSolver</i>	36
4.13	Diagram kelas <i>MinimumCameraPlacementSolverLPSolve</i>	37
B.1	Hasil 1	43
B.2	Hasil 2	43
B.3	Hasil 3	43
B.4	Hasil 4	43

DAFTAR TABEL

2.1	Kerangka tabel simplex	10
2.2	Tabel simplex awal	10
2.3	Pemilihan pivot	11
2.4	Pembaharuan tabel simplex	11
2.5	Tabel simplex iterasi ke-1	11
2.6	Tabel simplex iterasi ke-2	12
2.7	Tabel simplex iterasi ke-3	12
2.8	Tabel contoh	14
2.9	Tabel bewarna(1)	14
2.10	Tabel bewarna(2)	14

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Kamera merupakan alat/komponen optik yang digunakan untuk mengambil citra/gambar. Salah satu penggunaan kamera dalam kehidupan sehari-hari adalah kamera CCTV (*closed-circuit television*). Kamera CCTV digunakan untuk memantau suatu lokasi dengan tujuan pengawasan dan keamanan. Kamera CCTV pada umumnya dipasang pada tempat strategis sehingga memiliki tingkat jangkauan yang baik. Kamera CCTV bekerja dengan cara merekam lokasi dalam jangkauannya secara terus menerus dan menyimpan hasil rekamannya dalam media penyimpanan. Rekaman ini biasanya digunakan oleh petugas keamanan untuk memantau lokasi tersebut dari tempat yang berbeda sehingga petugas tidak perlu memantau lokasi tersebut dengan datang secara langsung. Petugas hanya perlu mendatangi lokasi tersebut apabila mendapati hal-hal yang mencurigakan berdasarkan hasil rekaman CCTV.

Penempatan kamera CCTV di ruangan yang berbentuk sederhana (persegi panjang) tidaklah sulit. Kamera CCTV yang dibutuhkan biasanya berjumlah dua buah dan dipasang di kedua sudut ruangan yang merupakan satu diagonal. Namun, jika ruangan berukuran besar, maka tujuan penggunaan kamera CCTV pun tidak hanya untuk mendeteksi adanya orang, tetapi juga mengenali orang tersebut. Hal ini tentunya menyebabkan kesulitan dalam menentukan jumlah minimum dan lokasi penempatan kamera CCTV. Terdapat beberapa pendekatan yang dapat digunakan untuk menyelesaikan masalah ini, seperti dengan cara memasang kamera CCTV pada daerah-daerah yang dapat dimasuki orang. Tetapi pada kasus terburuk, orang bisa saja masuk melewati jalur-jalur yang tidak diduga, seperti tembok, atap, bawah tanah, dsb. Oleh karena itu, alangkah baiknya pemasangan kamera CCTV dilakukan hingga seluruh daerah di ruangan tersebut tercakup sepenuhnya.

Penempatan kamera CCTV dapat dilakukan dalam berbagai lokasi dan berbagai arah pandang. Apabila penempatan kamera CCTV dilakukan tanpa adanya perhitungan, maka terdapat kemungkinan jumlah kamera yang terlalu banyak dan/atau seluruh lokasi yang tidak tercakup sepenuhnya. Pada penempatan kamera CCTV terdapat perhitungan tingkat *overlap* (penumpukan jangkauan) dan tingkat *out of bound* (jarak pandang terpotong). Penempatan kamera CCTV akan semakin baik apabila memiliki tingkat *overlap* dan tingkat *out of bound* yang semakin rendah.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang akan menghasilkan jumlah minimum beserta dengan penempatan-penempatan kamera CCTV berdasarkan ukuran ruangan, jarak pandang efektif kamera CCTV, dan sudut pandang kamera CCTV. Penempatan kamera CCTV akan terdiri atas lokasi penempatan dan juga arah pandangnya. Perangkat lunak juga akan memberikan visualisasi guna membantu pengguna memahami penempatan setiap kamera CCTV pada ruangan yang dimasukkan. Hasil dari perangkat lunak ini dapat dipastikan menjadi hasil yang paling optimal yang berarti bahwa hasil akan memiliki jumlah kamera CCTV yang paling minimum yang mencakup seluruh isi ruangan.

1.2 Rumusan Masalah

Berdasarkan deskripsi masalah yang dibahas sebelumnya, maka ditetapkan rumusan masalah sebagai berikut:

- Bagaimana cara menentukan jumlah minimum kamera CCTV dalam suatu ruangan?
- Bagaimana cara memvisualisasikan penempatan kamera-kamera CCTV dalam suatu ruangan?

1.3 Tujuan

Berdasarkan rumusan masalah, maka tujuan dalam skripsi ini adalah sebagai berikut:

- Mempelajari cara menentukan jumlah minimum kamera CCTV dalam suatu ruangan.
- Membangun perangkat lunak yang dapat memvisualisasikan penempatan kamera-kamera CCTV dalam suatu ruangan.

1.4 Batasan Masalah

Untuk mempermudah pembuatan template ini, tentu ada hal-hal yang harus dibatasi, misalnya saja bahwa template ini bukan berupa style \LaTeX pada umumnya (dengan alasannya karena belum mampu jika diminta membuat seperti itu)

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetur at, consectetur sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

1.5 Metodologi

Tentunya akan diisi dengan metodologi yang serius sehingga templatennya terkesan lebih serius.

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

1.6 Sistematika Pembahasan

Rencananya Bab 2 akan berisi petunjuk penggunaan template dan dasar-dasar \LaTeX . Mungkin bab 3,4,5 dapt diisi oleh ketiga jurusan, misalnya peraturan dasar skripsi atau pedoman penulisan, tentu jika berkenan. Bab 6 akan diisi dengan kesimpulan, bahwa membuat template ini ternyata sungguh menghabiskan banyak waktu.

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales

eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetur odio sem sed wisi.

BAB 2

LANDASAN TEORI

2.1 Linear Programming

Linear programming adalah teknik dalam ilmu matematika untuk menyelesaikan permasalahan yang berhubungan dengan optimasi. Teknik ini biasa digunakan dalam bidang produksi, manajemen operasional, penjadwalan, dan bidang-bidang lainnya yang melibatkan pengambilan keputusan. Pada bidang produksi, terdapat masalah ketika menentukan produk yang akan dihasilkan. Biasanya terdapat beberapa faktor yang menjadi pertimbangan seperti jumlah ketersediaan sumber daya, jumlah pekerja, dan besar keuntungan yang didapatkan. Dengan adanya faktor-faktor tersebut, penentuan terhadap produk yang akan dihasilkan harus dilakukan dengan perhitungan yang tepat. Apabila permasalahan tersebut dapat dimodelkan dengan tepat, maka masalah dapat diselesaikan dengan teknik linear programming sehingga menghasilkan solusi terbaik dengan memperhatikan setiap faktor yang ada.

2.1.1 Karakteristik

Masalah linear programming memiliki ciri-ciri sebagai berikut:

- Variabel keputusan

Di dalam masalah linear programming terdapat variabel-variabel keputusan. Variabel keputusan menunjukkan keputusan yang akan diambil.

$$x_1 = \text{jumlah produk A yang diproduksi}$$
$$x_2 = \text{jumlah produk B yang diproduksi}$$

- Fungsi Tujuan

Di dalam masalah linear programming terdapat tujuan yang akan dicapai. Fungsi tujuan menunjukkan tujuan yang akan dimaksimalkan atau diminimalkan.

$$\text{maximize } z = 3x_1 + 2x_2$$

- Batasan

Batasan dalam linear programming berfungsi untuk membatasi nilai variabel. Batasan juga dapat menunjukkan keterkaitan antar variabel keputusan.

$$2x_1 + x_2 \leq 100$$

$$x_1 + x_2 \leq 80$$

$$x_1 \leq 40$$

- Tanda Pembatas

Tanda pembatas berfungsi untuk menyatakan apakah variabel keputusan dapat bernilai negatif atau tidak. Jika variabel keputusan x_i tidak dapat bernilai negatif maka ditambahkan tanda pembatas $x_i \geq 0$ yang menyatakan bahwa variabel x_i tidak bernilai negatif. Namun, apabila nilai variabel keputusan tidak dibatasi atau dapat bernilai negatif, maka variabel tersebut disebut variabel bebas (*unrestricted in sign*).

$$x_1 \geq 0$$

$$x_2 \geq 0$$

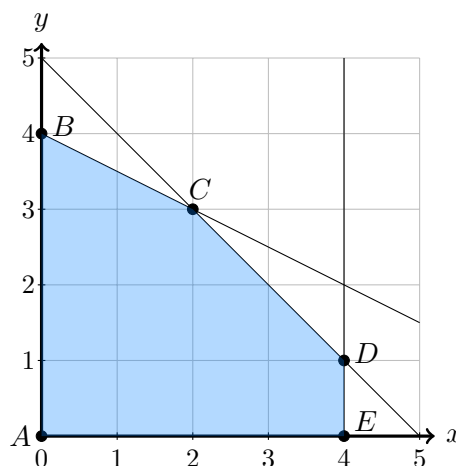
Dengan keempat karakteristik di atas, setiap masalah optimasi dapat dimodelkan ke bentuk masalah linear programming. Berikut contoh masalah linear programming:

$$\begin{array}{ll} \text{maximize} & z = 3x_1 + 2x_2 \\ \text{subject to} & 2x_1 + x_2 \leq 100 \\ & x_1 + x_2 \leq 80 \\ & x_1 \leq 40 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{array}$$

2.1.2 Daerah Solusi dan Solusi Optimal

Di dalam linear programming terdapat daerah solusi. Setiap batasan pada masalah linear programming akan menentukan daerah yang memenuhi batasan tersebut. Irisan dari gabungan daerah-daerah tersebut akan menghasilkan daerah solusi bagi masalah linear programming. Dengan adanya daerah solusi, maka masalah linear programming dapat diselesaikan karena memiliki solusi yang dibatasi.

Apabila masalah linear programming memiliki daerah solusi seperti pada Gambar 2.1, maka masalah linear programming tersebut memiliki solusi optimal. Daerah solusi pada setiap masalah linear programming berbentuk *convex polytope* yang terdiri dari banyak titik sudut. Solusi optimal terdapat pada salah satu titik sudut.

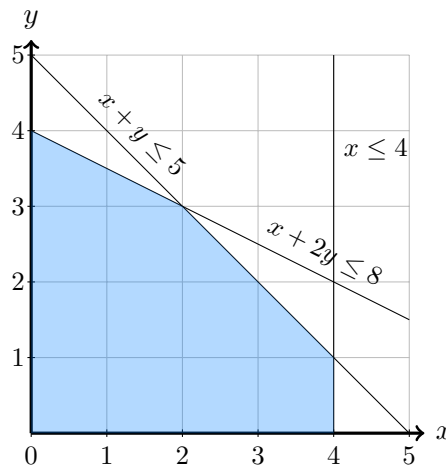


Gambar 2.1: Daerah solusi

Terdapat 3 jenis daerah solusi dalam masalah linear programming, yaitu:

1. Daerah solusi yang dibatasi (*bounded feasible region*)

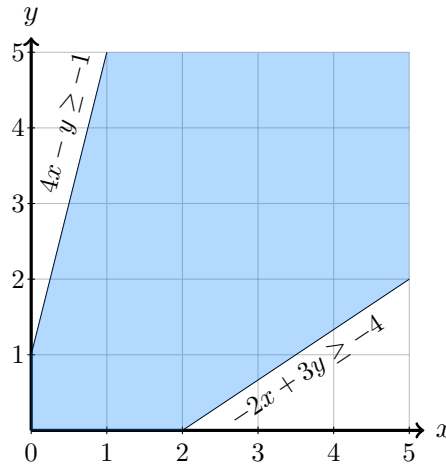
Batasan-batasan menghasilkan daerah solusi yang berbentuk *convex polytope*. Solusi optimal linear programming terdapat pada salah satu titik sudut. Gambar 2.2 berikut menggambarkan daerah solusi yang dibatasi.



Gambar 2.2: Daerah solusi yang dibatasi

2. Daerah solusi yang tidak dibatasi (*unbounded feasible region*)

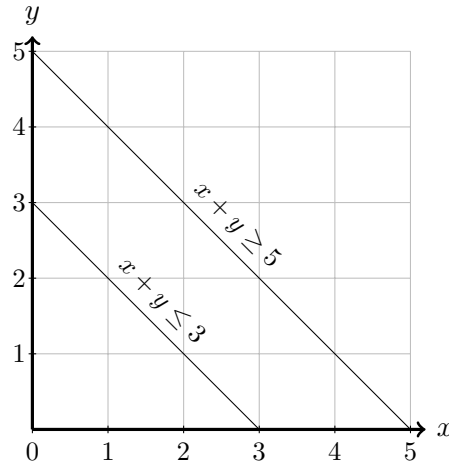
Batasan-batasan menghasilkan daerah solusi yang tidak tertutup. Daerah solusi ini memiliki jumlah solusi yang tak terhingga. Gambar 2.3 berikut menggambarkan daerah solusi yang tidak dibatasi.



Gambar 2.3: Daerah solusi yang tidak dibatasi

3. Tidak ada daerah solusi (*infeasible region*)

Batasan-batasan tidak membentuk daerah solusi sehingga masalah linear programming tidak memiliki solusi dan tidak dapat diselesaikan. Gambar 2.4 berikut menggambarkan tidak ada daerah solusi.



Gambar 2.4: Tidak ada daerah solusi

2.1.3 Bentuk Standar

Setiap model linear programming dapat dimodelkan ke dalam bentuk standar seperti berikut ini:

$$\begin{aligned}
 &\text{maximize} && c_1x_1 + c_2x_2 + \dots + c_nx_n \\
 &\text{subject to} && a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\
 &&& a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\
 &&& \vdots \\
 &&& a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m \\
 &&& x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0
 \end{aligned}$$

Bentuk tersebut merupakan bentuk standar masalah linear programming yang terdiri dari n buah variabel x dan m buah persamaan batasan. Variabel b_i, c_i, a_{ij} merupakan konstanta dan x_i merupakan variabel keputusan.

Bentuk standar tersebut dapat dinyatakan dalam notasi matriks seperti berikut ini:

$$\begin{aligned}
 &\text{maximize} && c^T x \\
 &\text{subject to} && Ax = b \text{ and } x \geq 0
 \end{aligned}$$

Pada notasi matriks, x adalah matriks kolom berdimensi n , C^T adalah matriks baris berdimensi n , A adalah matriks berdimensi $m \times n$, dan b adalah matriks kolom berdimensi m . Matriks $x \geq 0$ menunjukkan variabel keputusan x_i tidak dapat bernilai negatif.

Untuk mengubah ke bentuk standar, maka pertidaksamaan pada batasan harus diubah menjadi persamaan. Perubahan dibedakan menjadi 2 berdasarkan jenis tandanya, yaitu:

1. Pertidaksamaan lebih kecil (\leq)

Pertidaksamaan dengan tanda lebih kecil diubah ke bentuk persamaan dengan menambahkan variabel baru positif bernama *slack*. Lalu tambahkan tanda pembatas tidak negatif untuk variabel *slack* tersebut. Contoh:

$$x_1 + 2x_2 \leq 40$$

menjadi

$$\begin{aligned}
 x_1 + 2x_2 + s_1 &= 40 \\
 s_1 &\geq 0
 \end{aligned}$$

2. Pertidaksamaan lebih besar (\geq)

Pertidaksamaan dengan tanda lebih besar diubah ke bentuk persamaan dengan menambahkan variabel baru negatif bernama *surplus*. Lalu tambahkan tanda pembatas tidak negatif untuk variabel *surplus* tersebut. Contoh:

$$x_1 + 2x_2 \geq 40$$

menjadi

$$\begin{aligned} x_1 + 2x_2 - e_1 &= 40 \\ e_1 &\geq 0 \end{aligned}$$

2.1.4 Variabel Basis dan Non Basis

Masalah linear programming $Ax = b$ yang terdiri dari m buah persamaan batasan dan n buah variabel keputusan. Masalah linear programming $Ax = b$ memiliki solusi basis yang didapatkan dengan membuat $(n - m)$ buah variabel bernilai 0 dan menyelesaikan m buah variabel lainnya. Sebanyak $(n - m)$ buah variabel yang dibuat bernilai 0 disebut variabel non basis. Sedangkan sebanyak m buah variabel lainnya disebut variabel basis. Variabel basis merupakan variabel yang diselesaikan setelah membuat variabel lain menjadi variabel non basis. Berikut contoh sistem persamaan linear:

$$\begin{aligned} x_1 + x_2 &= 3 \\ x_2 + x_3 &= -1 \end{aligned}$$

Pada sistem persamaan linear di atas, dipilih sebanyak $3 - 2 = 1$ (3 variabel dan 2 persamaan) buah variabel yang akan menjadi variabel non basis. Jika himpunan variabel non basis $NBV = \{x_3\}$, maka himpunan variabel basis $BV = \{x_1, x_2\}$. Solusi dari sistem persamaan tersebut dapat dicari dengan membuat setiap variabel NBV menjadi variabel non basis dan menyelesaikan variabel BV.

$$\begin{aligned} x_1 + x_2 &= 3 \\ x_2 + 0 &= -1 \end{aligned}$$

Pada persamaan di atas didapatkan nilai $x_1 = 2$ dan $x_2 = 1$. Dengan demikian didapatkan solusi basis $x_1 = 2$, $x_2 = 1$, $x_3 = 0$.

2.1.5 Metode Simplex

Metode simplex merupakan metode yang digunakan untuk mencari solusi optimal dari masalah linear programming. Metode simplex bekerja secara beriterasi dengan menghasilkan solusi yang mendekati optimal pada setiap iterasinya. Iterasi ini akan dilakukan hingga tidak ditemukannya solusi yang lebih optimal. Berikut contoh masalah linear programming:

$$\begin{aligned} &\text{maximize} && 3x_1 + 2x_2 \\ &\text{subject to} && 2x_1 + x_2 \leq 18 \\ &&& 2x_1 + 3x_2 \leq 42 \\ &&& 3x_1 + x_2 \leq 24 \\ &&& x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

Langkah-langkah untuk menyelesaikan masalah linear programming dengan menggunakan metode simplex:

1. Mengubah masalah ke dalam bentuk standar

Masalah linear programming diubah ke bentuk standar sesuai dengan langkah-langkah yang sudah dibahas sebelumnya.

Berikut contoh masalah linear programming yang sudah diubah ke bentuk standar:

$$\begin{aligned}
 &\text{maximize} && 3x_1 + 2x_2 + 0x_3 + 0x_4 + 0x_5 \\
 &\text{subject to} && 2x_1 + x_2 + x_3 &= 18 \\
 &&& 2x_1 + 3x_2 &+ x_4 &= 42 \\
 &&& 3x_1 + x_2 &&+ x_5 &= 24 \\
 &&& x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0
 \end{aligned}$$

2. Membuat tabel *simplex*

Tabel 2.1 merupakan kerangka tabel simplex yang terdiri dari 4 kolom utama, yaitu kolom basis, kolom z , kolom variabel non basis, dan kolom ruas kanan/rhs (*right hand side*). Pada awalnya, baris z berisi nilai negatif dari konstanta variabel pada fungsi tujuan dan m baris berikutnya berisi konstanta variabel pada setiap persamaan batasan. m baris terakhir ini diidentifikasi sebagai baris basis.

Tabel 2.1: Kerangka tabel simplex

basic	z	x_1	x_2	\dots	x_n	rhs
z	1	$-c_1$	$-c_2$	\dots	$-c_n$	0
x_{n+1}	0	a_{11}	a_{12}	\dots	a_{1n}	b_1
x_{n+2}	0	a_{21}	a_{22}	\dots	a_{2n}	b_2
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots
x_{n+m}	0	a_{m1}	a_{m2}	\dots	a_{mn}	b_m

Tabel 2.2 berikut adalah tabel simplex awal untuk contoh masalah linear programming.

Tabel 2.2: Tabel simplex awal

basic	z	x_1	x_2	x_3	x_4	x_5	rhs
z	1	-3	-2	0	0	0	0
x_3	0	2	1	1	0	0	18
x_4	0	2	3	0	1	0	42
x_5	0	3	1	0	0	1	24

3. Mengecek solusi optimal

Apabila pada baris z tidak terdapat variabel dengan nilai negatif, maka solusi optimal sudah ditemukan dan menandakan akhir dari iterasi. Solusi optimal terdapat pada setiap rhs dari setiap baris basis. Jika masih terdapat variabel dengan nilai negatif, maka langkah selanjutnya adalah melakukan proses *pivoting*.

Pada Tabel 2.2, terdapat nilai negatif di baris z , yaitu pada variabel x_1 dan x_2 . Sehingga dilanjutkan ke proses *pivoting*.

4. Melakukan proses *pivoting*

Proses *pivoting* merupakan proses penukaran satu variabel basis dengan satu variabel non basis. Variabel basis yang akan digantikan (keluar dari baris basis) disebut dengan *leaving variable* dan variabel non basis yang akan menggantikan (masuk ke baris basis) disebut *entering variable*. Ketentuan dalam pemilihan variabel yang akan ditukar adalah sebagai berikut:

- *Entering variable* dipilih berdasarkan nilai terkecil dalam baris z . Kolom dari *entering variable* akan menjadi kolom pivot (*pivot column*).
- *Leaving variable* dipilih berdasarkan rasio tidak negatif terkecil antara rhs dengan nilai pada kolom pivot. Baris dari *leaving variable* akan menjadi baris pivot (*pivot row*).
- Pivot merupakan nilai yang berada di perpotongan kolom pivot dan baris pivot.

Pada Tabel 2.3, x_1 menjadi *entering variabel* karena bernilai paling kecil di baris z , yaitu -3 . Sedangkan baris basis x_5 menjadi *leaving variable* karena nilai rasionya yang paling kecil, yaitu 8. Perpotongan baris pivot dan kolom pivot menghasilkan pivot bernilai 3.

Tabel 2.3: Pemilihan pivot
pivot column

↓

basic	z	x_1	x_2	x_3	x_4	x_5	rhs	ratio:
z	1	-3	-2	0	0	0	0	$\frac{rhs_i}{pivotCol_i}$
x_3	0	2	1	1	0	0	18	$\frac{18}{2} = 9$
x_4	0	2	3	0	1	0	42	$\frac{42}{2} = 21$
x_5	0	(3)	1	0	0	1	24	$\frac{24}{3} = 8$

pivot row →

5. Memperbaharui tabel simplex

Setiap nilai pada baris basis baru akan dibagi dengan nilai pivot sebelumnya. Sedangkan nilai pada baris basis lainnya dikurangi dengan hasil perkalian antara nilai kolom pivot yang bersangkutan dengan nilai baru pada baris pivot yang bersangkutan. Setelah diperbaharui, langkah selanjutnya adalah kembali pada pengecekan solusi optimal.

Pada Tabel 2.4, x_1 menjadi baris basis yang baru. Setiap nilai pada baris tersebut dibagi dengan nilai pivot sebelumnya yang bernilai 3. Setiap nilai pada baris basis lainnya diubah sesuai dengan ketentuan yang sudah dibahas sebelumnya.

Tabel 2.4: Pembaharuan tabel simplex

basic	z	x_1	x_2	x_3	x_4	x_5	rhs
z	1	$-3 - (-3 \times \frac{3}{3})$	$-2 - (-3 \times \frac{1}{3})$	$0 - (-3 \times 0)$	$0 - (-3 \times 0)$	$0 - (-3 \times \frac{1}{3})$	$0 - (-3 \times \frac{24}{3})$
x_3	0	$2 - (2 \times \frac{3}{3})$	$1 - (2 \times \frac{1}{3})$	$1 - (2 \times 0)$	$0 - (2 \times 0)$	$0 - (2 \times \frac{1}{3})$	$18 - (2 \times \frac{24}{3})$
x_4	0	$2 - (2 \times \frac{3}{3})$	$3 - (2 \times \frac{1}{3})$	$0 - (2 \times 0)$	$1 - (2 \times 0)$	$0 - (2 \times \frac{1}{3})$	$42 - (2 \times \frac{24}{3})$
x_1	3	$\frac{3}{3}$	$\frac{1}{3}$	0	0	$\frac{1}{3}$	$\frac{24}{3}$

Tabel 2.5 berikut merupakan tabel simplex pada akhir iterasi ke-1:

Tabel 2.5: Tabel simplex iterasi ke-1

basic	z	x_1	x_2	x_3	x_4	x_5	rhs
z	1	0	-1	0	0	1	24
x_3	0	0	$\frac{1}{3}$	1	0	$-\frac{2}{3}$	2
x_4	0	0	$\frac{7}{3}$	0	1	$-\frac{2}{3}$	26
x_1	3	1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	8

Pada iterasi ke-1 masih terdapat nilai negatif pada baris z , sehingga iterasi masih dilakukan hingga tidak terdapat nilai negatif pada baris z . Tabel 2.6 dan Tabel 2.7 berikut merupakan hasil iterasi ke-2 dan ke-3.

Tabel 2.6: Tabel simplex iterasi ke-2

basic	z	x_1	x_2	x_3	x_4	x_5	rhs
z	1	0	0	3	0	-1	30
x_2	2	0	1	3	0	-2	6
x_4	0	0	0	-7	1	4	12
x_1	3	1	0	-1	0	1	6

Tabel 2.7: Tabel simplex iterasi ke-3

basic	z	x_1	x_2	x_3	x_4	x_5	rhs
z	1	0	0	$\frac{5}{4}$	$\frac{1}{4}$	0	33
x_2	2	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	12
x_5	0	0	0	$-\frac{7}{4}$	$\frac{1}{4}$	1	3
x_1	3	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	3

Pada iterasi ke-3 sudah tidak terdapat nilai negatif pada baris z . Dengan demikian, solusi optimal sudah dicapai, yaitu pada titik $x_1 = 3$ dan $x_2 = 12$.

2.2 Integer Programming

Integer programming merupakan teknik lanjutan dari linear programming yang dimana solusinya harus berupa bilangan bulat. Terdapat beberapa metode yang dapat digunakan untuk menyelesaikan masalah integer programming, salah satunya adalah branch and bound.

2.2.1 Branch and Bound

Branch and bound merupakan metode penyelesaian masalah integer programming dengan memecah masalah menjadi sub-sub masalah. Pemecahan masalah menjadi sub masalah akan terus dilakukan hingga didapatkan solusi berupa bilangan bulat. Berikut langkah-langkah metode branch and bound:

1. Selesaikan masalah menggunakan linear programming. Apabila solusi yang didapat berupa bilangan bulat, maka proses berhenti pada tahap ini. Solusi ini menjadi solusi akhir bagi permasalahan ini. Apabila sebaliknya, maka dilanjutkan ke tahap selanjutnya. Solusi pada tahap ini akan menjadi solusi batas atas yang akan digunakan sebagai pembanding di tahap-tahap selanjutnya.
2. Pecah masalah menjadi 2 sub masalah. Sub-sub masalah merupakan masalah yang sama dengan masalah sebelum dipecah namun memiliki sebuah tambahan batasan baru. Seelum menambah batasan baru tersebut, tentukan satu variabel keputusan dari masalah sebelum dipecah yang solusinya tidak berupa bilangan bulat. Lakukan pembulatan bilangan ke atas dan ke bawah terhadap solusi dari variabel keputusan tersebut. Lalu pada sub masalah pertama, tambahkan batasan untuk variabel keputusan tersebut agar bernilai lebih kecil daripada hasil pembulatan ke bawah. Sedangkan pada sub masalah kedua, tambahkan batasan untuk variabel keputusan tersebut agar bernilai lebih besar daripada hasil pembulatan ke atas.
3. Selesaikan sub masalah menggunakan linear programming dan periksa solusinya. Tahap ini dibedakan menjadi 3 berdasarkan solusinya, yaitu:
 - Solusi berupa bilangan bulat

Apabila solusi berupa bilangan bulat, maka bandingkan solusi ini dengan solusi batas atas. Apabila solusi ini sama dengan batas atas, maka solusi ini merupakan solusi akhir bagi permasalahan ini dan proses berhenti pada tahap ini. Apabila solusi lebih buruk dari solusi batas atas, maka bandingkan dengan solusi batas bawah. Apabila solusi ini lebih baik daripada solusi batas bawah, maka jadikanlah solusi ini sebagai solusi batas bawah yang baru.

- Solusi tidak berupa bilangan bulat

Apabila solusi tidak berupa bilangan bulat, maka periksa apakah solusi ini lebih baik daripada solusi batas bawah. Apabila lebih baik, maka lakukan tahap ke-2 untuk memecah sub masalah ini. Apabila tidak lebih baik, maka sub masalah ini tidak perlu dipecah karena tidak akan menghasilkan solusi yang lebih baik.

- Sub masalah tidak dapat diselesaikan

Sub masalah ini tidak memiliki solusi sehingga tidak perlu dilanjutkan lagi.

4. Periksa apakah terdapat sub masalah yang belum diselesaikan. Apabila masih terdapat sub masalah yang belum diselesaikan, maka kembali ke tahap ke-3 untuk menyelesaikannya. Apabila seluruh sub masalah telah diselesaikan, maka lanjutkan ke tahap selanjutnya.
5. Periksa solusi batas bawah. Apabila terdapat solusi batas bawah, maka solusi tersebut merupakan solusi akhir bagi permasalahan ini. Apabila tidak terdapat solusi batas bawah, maka permasalahan ini tidak memiliki solusi yang berupa bilangan bulat

2.3 Template Skripsi FTIS UNPAR

Akan dipaparkan bagaimana menggunakan template ini, termasuk petunjuk singkat membuat referensi, gambar dan tabel. Juga hal-hal lain yang belum terpikir sampai saat ini.

Nulla in ipsum. Praesent eros nulla, congue vitae, euismod ut, commodo a, wisi. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nonummy magna non leo. Sed felis erat, ullamcorper in, dictum non, ultricies ut, lectus. Proin vel arcu a odio lobortis euismod. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Proin ut est. Aliquam odio. Pellentesque massa turpis, cursus eu, euismod nec, tempor congue, nulla. Duis viverra gravida mauris. Cras tincidunt. Curabitur eros ligula, varius ut, pulvinar in, cursus faucibus, augue.

Nulla mattis luctus nulla. Duis commodo velit at leo. Aliquam vulputate magna et leo. Nam vestibulum ullamcorper leo. Vestibulum condimentum rutrum mauris. Donec id mauris. Morbi molestie justo et pede. Vivamus eget turpis sed nisl cursus tempor. Curabitur mollis sapien condimentum nunc. In wisi nisl, malesuada at, dignissim sit amet, lobortis in, odio. Aenean consequat arcu a ante. Pellentesque porta elit sit amet orci. Etiam at turpis nec elit ultricies imperdiet. Nulla facilisi. In hac habitasse platea dictumst. Suspendisse viverra aliquam risus. Nullam pede justo, molestie nonummy, scelerisque eu, facilisis vel, arcu.

2.3.1 Tabel

Berikut adalah contoh pembuatan tabel. Penempatan tabel dan gambar secara umum diatur secara otomatis oleh \LaTeX , perhatikan contoh di file bab2.tex untuk melihat bagaimana cara memaksa tabel ditempatkan sesuai keinginan kita.

Perhatikan bawa berbeda dengan penempatan judul gambar gambar, keterangan tabel harus diletakkan di atas tabel!! Lihat Tabel 2.8 berikut ini:

Tabel 2.8: Tabel contoh

	v_{start}	\mathcal{S}_1	v_{end}
τ_1	1	12	20
τ_2	1		20
τ_3	1	9	20
τ_4	1		20

Tabel 2.9 dan Tabel 2.10 berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 2.9: Tabel bewarna(1)

	v_{start}	\mathcal{S}_2	\mathcal{S}_1	v_{end}
τ_1	1	5	12	20
τ_2	1	8		20
τ_3	1	2/8/17	9	20
τ_4	1			20

Tabel 2.10: Tabel bewarna(2)

	v_{start}	\mathcal{S}_1	\mathcal{S}_2	v_{end}
τ_1	1	12	5	20
τ_2	1		8	20
τ_3	1	9	2/8/17	20
τ_4	1			20

2.3.2 Kutipan

Berikut contoh kutipan dari berbagai sumber, untuk keterangan lebih lengkap, silahkan membaca file referensi.bib yang disediakan juga di template ini. Contoh kutipan:

- Buku: [1]
- Bab dalam buku: [2]
- Artikel dari Jurnal: [3]
- Artikel dari prosiding seminar/konferensi: [4]
- Skripsi/Thesis/Disertasi: [5] [6] [7]
- Technical/Scientific Report: [8]
- RFC (Request For Comments): [9]
- Technical Documentation/Technical Manual: [10] [11] [12]
- Paten: [13]
- Tidak dipublikasikan: [14] [15]
- Laman web: [16]
- Lain-lain: [17]

2.3.3 Gambar

Pada hampir semua editor, penempatan gambar di dalam dokumen \LaTeX tidak dapat dilakukan melalui proses *drag and drop*. Perhatikan contoh pada file bab2.tex untuk melihat bagaimana cara menempatkan gambar. Beberapa hal yang harus diperhatikan pada saat menempatkan gambar:

- Setiap gambar **harus** diacu di dalam teks (gunakan *field* LABEL)

- *Field* CAPTION digunakan untuk teks pengantar pada gambar. Terdapat dua bagian yaitu yang ada di antara tanda [dan] dan yang ada di antara tanda { dan }. Yang pertama akan muncul di Daftar Gambar, sedangkan yang kedua akan muncul di teks pengantar gambar. Untuk skripsi ini, samakan isi keduanya.
- Jenis file yang dapat digunakan sebagai gambar cukup banyak, tetapi yang paling populer adalah tipe PNG (lihat Gambar 2.5), tipe JPG (Gambar 2.6) dan tipe PDF (Gambar 2.7)
- Besarnya gambar dapat diatur dengan *field* SCALE.
- Penempatan gambar diatur menggunakan *placement specifier* (di antara tanda [dan] setelah deklarasi gambar. Yang umum digunakan adalah **H** untuk menempatkan gambar **sesuai** penempatannya di file .tex atau **h** yang berarti "kira-kira" di sini. Jika tidak menggunakan *placement specifier*, L^AT_EX akan menempatkan gambar secara otomatis untuk menghindari bagian kosong pada dokumen anda. Walaupun cara ini sangat mudah, hindarkan terjadinya penempatan dua gambar secara berurutan.
 - Gambar 2.5 ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
 - Gambar 2.6 dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisannya di dalam file bab2.tex!
 - Gambar 2.7 ditempatkan menggunakan *specifier* **h**.

Curabitur tellus magna, porttitor a, commodo a, commodo in, tortor. Donec interdum. Praesent scelerisque. Maecenas posuere sodales odio. Vivamus metus lacus, varius quis, imperdiet quis, rhoncus a, turpis. Etiam ligula arcu, elementum a, venenatis quis, sollicitudin sed, metus. Donec nunc pede, tincidunt in, venenatis vitae, faucibus vel, nibh. Pellentesque wisi. Nullam malesuada. Morbi ut tellus ut pede tincidunt porta. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam congue neque id dolor.

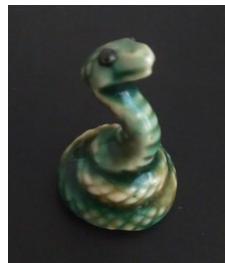
Donec et nisl at wisi luctus bibendum. Nam interdum tellus ac libero. Sed sem justo, laoreet vitae, fringilla at, adipiscing ut, nibh. Maecenas non sem quis tortor eleifend fermentum. Etiam id tortor ac mauris porta vulputate. Integer porta neque vitae massa. Maecenas tempus libero a libero posuere dictum. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aenean quis mauris sed elit commodo placerat. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Vivamus rhoncus tincidunt libero. Etiam elementum pretium justo. Vivamus est. Morbi a tellus eget pede tristique commodo. Nulla nisl. Vestibulum sed nisl eu sapien cursus rutrum.

Nulla non mauris vitae wisi posuere convallis. Sed eu nulla nec eros scelerisque pharetra. Nullam varius. Etiam dignissim elementum metus. Vestibulum faucibus, metus sit amet mattis rhoncus, sapien dui laoreet odio, nec ultricies nibh augue a enim. Fusce in ligula. Quisque at magna et nulla commodo consequat. Proin accumsan imperdiet sem. Nunc porta. Donec feugiat mi at justo. Phasellus facilisis ipsum quis ante. In ac elit eget ipsum pharetra faucibus. Maecenas viverra nulla in massa.

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.



Gambar 2.5: Gambar *Serpentes* dalam format png



Gambar 2.6: Ular kecil

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetur quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.



Gambar 2.7: *Serpentes* jantan

BAB 3

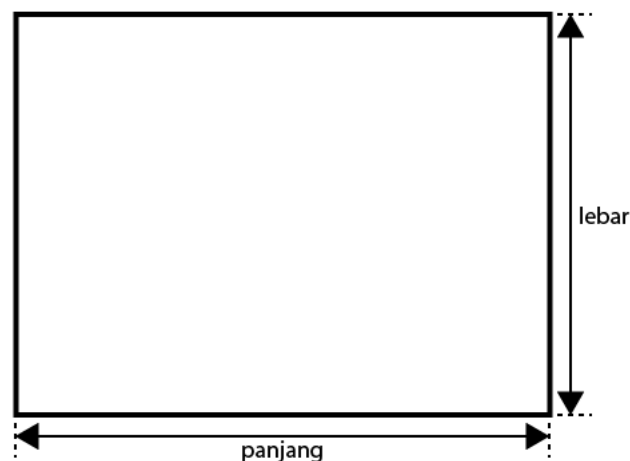
ANALISIS

3.1 Pemodelan Masalah

Masalah yang dibahas di skripsi ini perlu dirumuskan terlebih dahulu agar dapat diselesaikan. Masalah akan dipecah menjadi beberapa elemen sehingga fungsi dari setiap elemen dapat dipahami lebih mudah. Setiap elemen akan memiliki keterhubungan satu dengan yang lainnya sehingga apabila disatukan akan merepresentasikan masalah yang dibahas. Dengan merumuskan masalah, maka masalah dapat dimodelkan menjadi elemen-elemen yang dapat dipahami secara konkret baik bagi penulis maupun pembaca.

3.1.1 Ruangan

Dalam masalah, terdapat sebuah ruangan yang harus dicakup sepenuhnya oleh kamera-kamera CCTV. Ruangan dapat diartikan sebagai sebuah bidang 3 dimensi yang memiliki rongga di dalamnya. Ruangan ini pada umumnya memiliki bentuk yang beragam sesuai dengan arsitekturnya pada saat dibangun. Berbeda dengan ruangan tersebut, ruangan yang dibahas dalam masalah ini memiliki ukuran dimensi dan bentuk yang dibatasi. Ruangan tidak dimodelkan dalam bentuk 3 dimensi, melainkan dalam bidang 2 dimensi yang berbentuk persegi panjang. Dengan pemodelan ini, ruangan akan memiliki 2 parameter utama yang menentukan ukuran ruangan, yaitu ukuran panjang dan ukuran lebar. Ukuran panjang dan ukuran lebar ini memiliki satuan berupa sentimeter(cm). Pemodelan ruangan dapat dipahami lebih lanjut pada gambar 3.1.

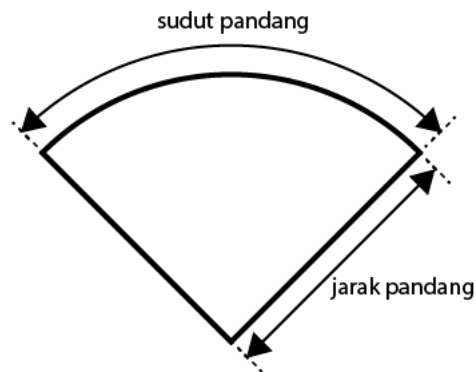


Gambar 3.1: Pemodelan ruangan

3.1.2 Kamera CCTV

Kamera CCTV yang beredar di pasaran memiliki spesifikasi yang sangat beragam. Dalam masalah ini, jenis kamera CCTV akan dibatasi sehingga hanya berjumlah 1 buah saja. Kamera CCTV

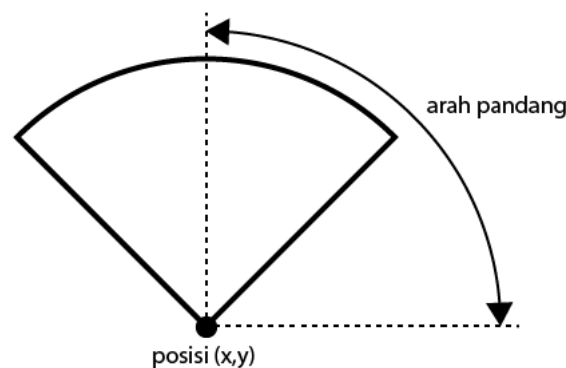
sendiri memiliki berbagai parameter seperti jarak pandang, lebar sudut pandang, tingkat resolusi, dan parameter-parameter lainnya. Dalam masalah ini, terdapat 2 parameter utama yang digunakan, yaitu jarak pandang efektif dan lebar sudut pandang. Jarak pandang efektif merupakan jarak pandang terjauh kamera CCTV untuk mengenali suatu objek yang akan dipantau. Jarak pandang efektif dinyatakan dalam ukuran bersatuan sentimeter(cm) dan lebar sudut pandang dinyatakan dalam ukuran derajat. Pemodelan kamera CCTV dapat dipahami lebih lanjut pada gambar 3.2.



Gambar 3.2: Pemodelan kamera CCTV

3.1.3 Penempatan Kamera CCTV

Setiap kamera CCTV dapat ditempatkan di mana saja selama berada di dalam ruangan. Penempatan kamera CCTV terdiri dari 2 komponen utama, yaitu posisi penempatan dan arah pandang. Posisi dan arah pandang akan mempengaruhi daerah cakupan kamera CCTV yang bersangkutan. Posisi penempatan dimodelkan dengan menggunakan sistem koordinat kartesius sehingga dapat dinyatakan dalam bentuk koordinat (x,y) . Sumbu y pada sistem koordinat yang digunakan akan dibalik agar sesuai dengan lingkungan grafis pada layar komputer. Arah pandang kamera CCTV dinyatakan sebagai besar sudut perpotongan antara garis tengah kamera CCTV dengan garis 0° yang dituliskan dalam satuan derajat. Dengan demikian, penempatan kamera CCTV terdiri atas posisi penempatan dan arah pandang yang dituju. Pemodelan penempatan kamera CCTV dapat dipahami lebih lanjut pada gambar 3.3.



Gambar 3.3: Pemodelan penempatan kamera CCTV

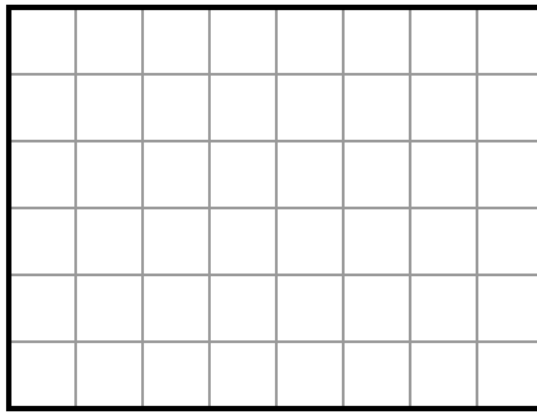
3.1.4 Daerah Cakupan

Daerah cakupan kamera CCTV memiliki bentuk yang tidak sederhana sehingga menjadi sulit ketika akan diolah. Terdapat 3 kasus yang menjelaskan daerah cakupan dengan bentuk yang tidak

sederhana ini, yaitu:

1. Kasus yang terjadi ketika penempatan kamera-kamera CCTV yang dilakukan dengan sedemikian rupa sehingga menghasilkan suatu daerah yang memiliki bentuk yang tidak sederhana.
2. Kasus yang terjadi ketika terdapat 2 atau lebih penempatan kamera CCTV yang mencakup suatu daerah yang sama sehingga daerah irisan tersebut memiliki bentuk yang tidak sederhana.
3. Kasus yang terjadi ketika suatu penempatan kamera CCTV mencakup daerah di luar ruangan sehingga daerah tersebut memiliki bentuk yang tidak sederhana.

Dengan adanya ketiga kasus ini, maka daerah cakupan perlu didefinisikan dan dimodelkan lebih lanjut agar kasus tersebut dapat dihindari. Ruangan dapat dimodelkan lebih lanjut sehingga berbentuk grid point seperti pada gambar 3.4. Grid point akan memecah ruangan ke dalam bagian-bagian yang lebih kecil yang disebut dengan cell.



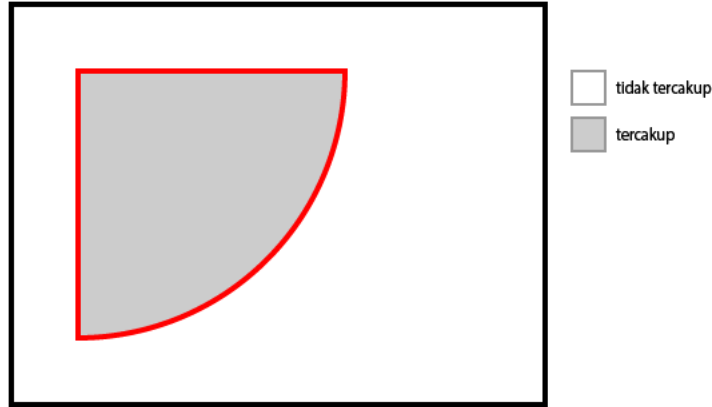
Gambar 3.4: Pemodelan ruangan dalam bentuk grid point

Cell tidak selalu berbentuk persegi, namun dapat berbentuk persegi panjang. Hal ini dikarenakan susunan cell-cell harus menghasilkan ukuran yang sama dengan ukuran ruangan. Untuk menentukan ukuran cell, ditentukan sebuah ukuran yang menyatakan ukuran terbesar yang dapat dimiliki cell. Ukuran ini akan digunakan untuk menentukan ukuran cell yang apabila disusun akan menghasilkan ukuran ruangan. Berikut ini merupakan rumus yang digunakan untuk mencari ukuran cell:

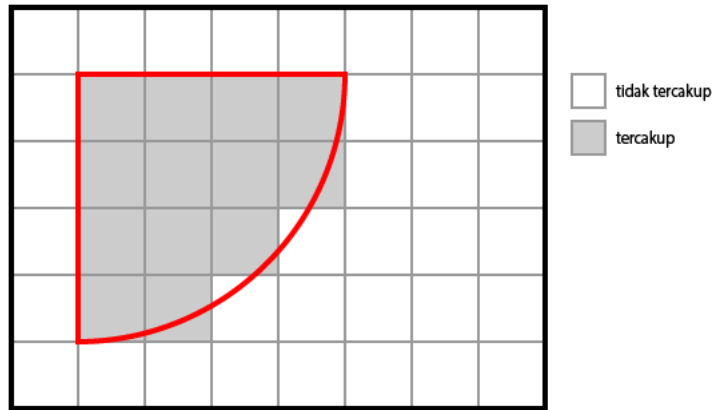
$$\begin{aligned}
 \text{columns} &= \left\lceil \frac{\text{room width}}{\text{max cell size}} \right\rceil \\
 \text{rows} &= \left\lceil \frac{\text{room length}}{\text{max cell size}} \right\rceil \\
 \text{cell width} &= \frac{\text{room width}}{\text{columns}} \\
 \text{cell length} &= \frac{\text{room length}}{\text{rows}}
 \end{aligned}$$

Dengan rumus ini, akan didapatkan ukuran panjang dan ukuran lebar cell yang tidak melebihi ukuran terbesar cell. Apabila ukuran cell ini dikalikan dengan jumlah kolom dan jumlah baris grid point, maka akan didapatkan ukuran yang sama dengan ukuran ruangan.

Setiap cell berfungsi merepresentasikan sebagian daerah dalam ruangan yang berupa satu kesatuan sehingga kasus bentuk daerah yang tidak sederhana dapat dihindari. Dengan pemodelan menggunakan grid point, maka daerah cakupan dari suatu kamera CCTV dapat dinyatakan dalam bentuk kumpulan cell. Gambar 3.5 dan gambar 3.6 menunjukkan perbandingan daerah cakupan kamera CCTV ketika sebelum dan sesudah dimodelkannya ruangan dalam bentuk grid point.



Gambar 3.5: Daerah cakupan sebelum pemodelan grid point



Gambar 3.6: Daerah cakupan sesudah pemodelan grid point

Untuk mencari kumpulan cell yang dicakup suatu penempatan kamera CCTV, maka setiap cell harus diperiksa apakah dapat tercakup oleh penempatan tersebut. Pada setiap cell akan ditentukan sebuah titik tengah yang berada di tengah-tengah cell. Titik tengah ini akan digunakan dalam pemeriksaan ketercakupan cell. Pemeriksaan terdiri dari pemeriksaan jarak cell dan pemeriksaan sudut rotasi cell. Jarak antara titik tengah cell dengan titik penempatan kamera CCTV harus lebih kecil daripada jarak pandang kamera CCTV. Sudut rotasi cell harus berada di antara sudut pandang kamera CCTV. Terdapat rumus yang digunakan untuk mendapatkan sudut rotasi ini, yaitu sebagai berikut:

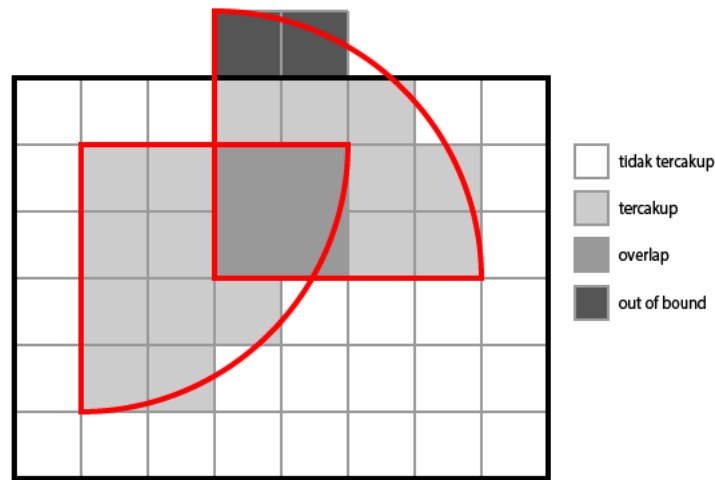
$$\text{atan2}(x, y) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$

$$\theta = \text{atan2}(y_{cam} - y_{cell}, x_{cell} - x_{cam})$$

Pada rumus tersebut, titik (x_{cell}, y_{cell}) menunjukkan titik tengah cell dan titik (x_{cam}, y_{cam}) menunjukkan titik penempatan kamera CCTV. Sudut rotasi cell (θ) akan dibandingkan dengan sudut mulai dan sudut akhir dari sudut pandang kamera CCTV. Sudut rotasi harus berada di antara kedua sudut tersebut. Apabila kedua pemeriksaan tersebut berhasil dilalui, maka cell dinyatakan

tercakup oleh kamera CCTV. Apabila sebaliknya, maka cell dinyatakan tidak tercakup oleh kamera CCTV. Dengan pemeriksaan ini, maka cakupan kamera CCTV dapat dicari dan dinyatakan dalam bentuk kumpulan cell.

Dengan pemodelan grid point, perhitungan tingkat *overlap* dan *out of bound* dapat dilakukan dengan membandingkan jumlah cell. *Overlap cell* adalah cell yang dicakup oleh lebih dari 1 kamera CCTV. *Out of bound cell* adalah cell yang tercakup oleh kamera CCTV, tetapi berada di luar ruangan. Gambar 3.7 menggambarkan *overlap cell* dan *out of bound cell*. Tingkat *overlap* dan *out of bound* dapat dihitung dengan mengtotalkan jumlah cell yang dicakup dari setiap kamera CCTV dan membaginya dengan jumlah cell yang berada di dalam ruangan. Perhitungan tingkat *overlap* dan *out of bound* hanya bisa dilakukan apabila seluruh cell dalam ruangan telah tercakup sepenuhnya.



Gambar 3.7: Daerah overlap dan out of bound

3.2 Penyelesaian Masalah

Dalam masalah ini, terdapat tujuan yang akan dicapai, yaitu mendapatkan penempatan-penempatan kamera CCTV yang dapat mencakup seluruh daerah pada ruangan dengan maksimal tingkat overlap dan out of bound sebesar 35%. Kamera-kamera CCTV tentu dapat ditempatkan dimana saja hingga seluruh daerah pada ruangan tercakup sepenuhnya. Namun penempatan-penempatan ini tidak selalu menghasilkan tingkat overlap dan out of bound yang bernilai maksimal sebesar 35%. Cara ini tentu tidak efektif sehingga diperlukan metode lainnya untuk menyelesaikan masalah ini. Salah satu cara yang digunakan untuk menyelesaikan masalah ini adalah dengan menggunakan teknik linear programming.

3.2.1 Variabel Keputusan

Pada awalnya ditentukan seluruh kemungkinan penempatan kamera CCTV sehingga setiap cell pada ruangan dicakup oleh minimal 1 kamera CCTV. Setiap penempatan kamera CCTV memiliki posisi dan arah pandang masing-masing. Dari seluruh kemungkinan ini akan dicari himpunan bagian yang dimana penempatan-penempatannya dapat mencakup seluruh daerah pada ruangan. Setiap kemungkinan penempatan memiliki 2 kemungkinan, yaitu diterapkan sebagai bagian dari solusi atau tidak. Dalam penyelesaian menggunakan teknik linear programming, setiap kemungkinan penempatan akan menjadi variabel keputusan. Variabel keputusan $x_{ij\theta}$ akan merujuk pada penempatan kamera CCTV pada posisi (i, j) dengan arah pandang θ . Karena setiap penempatan kamera CCTV memiliki 2 kemungkinan untuk diterapkan, maka setiap variabel keputusan dapat dinyatakan dengan nilai 0

atau 1. Apabila suatu variabel keputusan bernilai 1, maka penempatan kamera CCTV yang dirujuk akan diterapkan sebagai bagian dari solusi. Apabila bernilai 0, maka penempatan kamera CCTV yang dirujuk tidak akan diterapkan. Karena variabel keputusan hanya bisa bernilai 0 dan 1 saja, masalah ini tidak diselesaikan menggunakan teknik linear programming biasa karena hasil dapat bernilai pecahan. Masalah ini perlu diselesaikan menggunakan teknik integer programming agar hasil dari setiap variabel keputusan hanya berupa bilangan bulat. Variabel keputusan dinyatakan dalam notasi berikut:

$$x_{ij\theta} = \begin{cases} 1 & \text{jika kamera CCTV ditempatkan pada posisi } (i, j) \\ & \text{dengan arah pandang } \theta \\ 0 & \text{jika kamera CCTV tidak ditempatkan} \end{cases}$$

3.2.2 Fungsi Tujuan

Solusi yang diharapkan dari masalah ini adalah mendapatkan penempatan-penempatan kamera CCTV yang paling minimum untuk mencakup seluruh daerah dalam ruangan. Dengan semakin sedikitnya penempatan-penempatan kamera CCTV yang digunakan, tingkat overlap dan out of bound juga akan semakin kecil. Sebelumnya telah diketahui bahwa setiap variabel keputusan hanya bisa bernilai 0 atau 1 saja. Jumlah penempatan kamera CCTV pun dapat dicari dengan menjumlahkan seluruh variabel keputusan. Fungsi tujuan dalam linear programming dapat dinyatakan sebagai jumlah penempatan kamera CCTV akan diminimumkan. Fungsi tujuan dituliskan dalam notasi berikut:

$$\text{minimize } z = \sum_{\theta=0}^{s_{\theta}-1} \sum_{i=0}^{s_i-1} \sum_{j=0}^{s_j-1} x_{ij\theta}$$

3.2.3 Batasan

Selain mencari jumlah kamera CCTV yang minimal, seluruh daerah pada ruangan harus tercakup sepenuhnya. Berdasarkan pemodelan, ruangan dimodelkan dalam bentuk grid point sehingga daerah pada ruangan dimodelkan dalam bentuk cell. Agar seluruh daerah pada ruangan tercakup sepenuhnya, maka setiap cell harus dicakup oleh minimal 1 kamera CCTV. Terdapat sebuah fungsi biner yang digunakan untuk mengetahui apakah suatu penempatan kamera CCTV dapat mencakup suatu cell. Fungsi biner ini akan menghasilkan nilai 1 apabila suatu penempatan kamera CCTV dapat mencakup suatu cell. Apabila sebaliknya, maka fungsi biner ini akan menghasilkan nilai 0. Fungsi tersebut ditulis sebagai berikut:

$$cov(i, j, \theta, p, q) = \begin{cases} 1 & \text{jika kamera CCTV pada posisi } (i, j) \text{ dengan arah pandang } \theta \\ & \text{dapat mencakup cell } (p, q) \\ 0 & \text{jika sebaliknya} \end{cases}$$

Dalam bentuk linear programming akan ditambahkan batasan yang menyatakan bahwa setiap cell harus dicakup oleh minimal 1 penempatan kamera CCTV. Fungsi biner sebelumnya akan digunakan untuk menyatakan hubungan ketercukupan cell dengan penempatan kamera CCTV. Batasan tersebut ditulis sebagai berikut:

$$\sum_{\theta=0}^{s_{\theta}-1} \sum_{i=0}^{s_i-1} \sum_{j=0}^{s_j-1} x_{i,j,\theta} \times cov(i, j, \theta, p, q) \geq 1$$

$$0 \leq p \leq (s_p - 1), 0 \leq q \leq (s_q - 1)$$

3.2.4 Bentuk Linear Programming

Masalah yang dibahas di dalam skripsi ini dapat diubah ke dalam bentuk yang dapat diselesaikan dengan teknik linear programming. Setiap penempatan kamera CCTV pada ruangan menjadi variabel-variabel keputusan yang menunjukkan apakah akan diterapkan sebagai solusi atau tidak. Fungsi tujuan berfungsi untuk menyatakan bahwa penempatan-penempatan kamera CCTV harus berjumlah minimum sehingga sesuai dengan solusi yang diharapkan. Selain mendapatkan jumlah penempatan kamera CCTV yang minimum, terdapat batasan-batasan yang menyatakan bahwa seluruh daerah pada ruangan juga harus tercakup sepenuhnya. Apabila digabungkan, maka bentuk masalah ini dalam bentuk linear programming adalah seperti berikut:

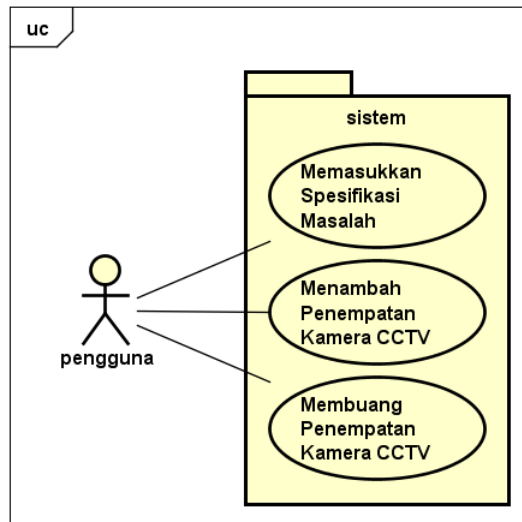
$$\begin{aligned}
 & \text{minimize } z = \sum_{\theta=0}^{s_{\theta}-1} \sum_{i=0}^{s_i-1} \sum_{j=0}^{s_j-1} x_{ij\theta} \\
 & \text{subject to } \sum_{\theta=0}^{s_{\theta}-1} \sum_{i=0}^{s_i-1} \sum_{j=0}^{s_j-1} x_{i,j,\theta} \times \text{cov}(i, j, \theta, p, q) \geq 1 \\
 & 0 \leq p \leq (s_p - 1), 0 \leq q \leq (s_q - 1) \\
 & x_{ij\theta} \in \{0, 1\}
 \end{aligned}$$

3.3 Analisis Kebutuhan Perangkat Lunak

Pada subbab ini, akan dijelaskan aksi-aksi yang dapat dilakukan pengguna terhadap perangkat lunak melalui diagram *use case* dan sekanrio-skenario. Selain penjelasan aksi-aksi, terdapat juga diagram kelas sederhana yang akan dikembangkan lebih lanjut pada tahap perancangan perangkat lunak.

3.3.1 Diagram Use Case

Dalam perangkat lunak yang dibangun, hanya terdapat 1 jenis pengguna. Diagram *use case* pada gambar 3.8 menunjukkan aktor dan aksi-aksi yang dapat dilakukannya.



Gambar 3.8: Diagram *use case*

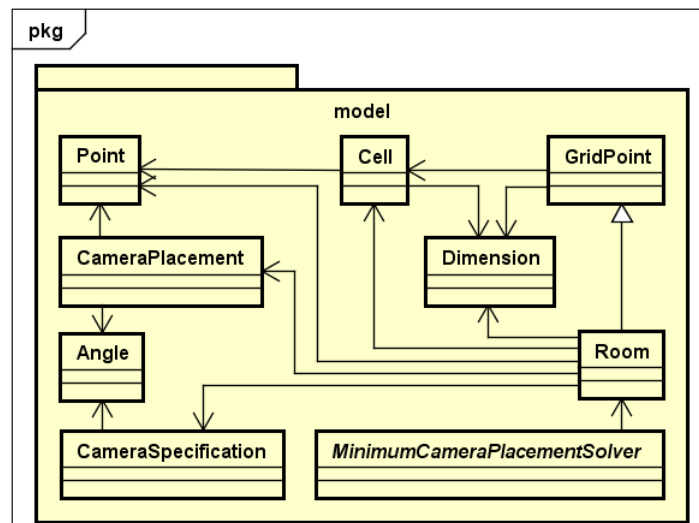
Berikut skenario dari setiap aksi pada diagram *use case*:

- Skenario: **Memasukkan Spesifikasi Masalah**

- Aktor: Pengguna
- Langkah:
 1. Aktor memasukkan spesifikasi masalah yang terdiri dari ukuran ruangan dan spesifikasi kamera CCTV dan dilanjutkan dengan menekan tombol "submit".
 2. Sistem menampilkan tampilan simulasi penempatan kamera CCTV.
- Skenario: **Menambah Penempatan Kamera CCTV**
 - Aktor: Pengguna
 - Langkah:
 1. Aktor memilih posisi penempatan kamera CCTV dan mengisi sudut arah pandang penempatan kamera CCTV dan dilanjutkan dengan menekan tombol "add camera".
 2. Sistem menambahkan penempatan ke dalam simulasi dan memperbaharui tampilan simulasi.
- Skenario: **Membuang Penempatan Kamera CCTV**
 - Aktor: Pengguna
 - Langkah:
 1. Aktor memilih penempatan kamera CCTV yang akan dibuang dan menekan tombol "remove".
 2. Sistem membuang penempatan dari simulasi dan memperbaharui tampilan simulasi.

3.3.2 Diagram Kelas

Pada bagian ini terdapat diagram kelas sederhana yang menunjukkan kelas-kelas yang akan digunakan untuk merancang perangkat lunak. Diagram kelas sederhana dapat dilihat pada gambar 3.9.



Gambar 3.9: Diagram kelas sederhana

BAB 4

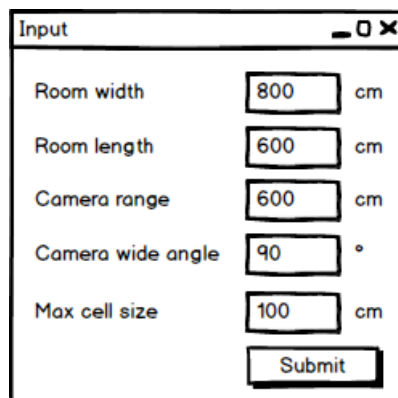
PERANCANGAN

4.1 Perancangan Antarmuka

Dalam perangkat lunak yang dibangun, tampilan yang digunakan adalah tampilan antarmuka grafis. Tampilan antarmuka ini berguna untuk mempermudah interaksi pengguna dengan perangkat lunak. Selain mempermudah, tampilan ini digunakan agar dapat menghasilkan visualisasi penempatan kamera CCTV sehingga pengguna dapat memahami penempatan-penempatan tersebut. Pada bagian ini akan dijelaskan bentuk dari setiap antarmuka. Berikut bentuk antarmuka-antarmuka tersebut:

1. Antarmuka: **Penerima Masukan**

Antarmuka ini berfungsi untuk menerima masukan dari pengguna. Antarmuka ini dapat dilihat pada gambar 4.1. Pada antarmuka ini terdapat kolom-kolom masukan yang dapat diisi oleh pengguna. Pengguna dapat mengisi ukuran ruangan, spesifikasi kamera CCTV, dan ukuran terbesar cell pada kolom-kolom tersebut. Apabila pengguna sudah yakin dengan masukannya, maka pengguna dapat menekan tombol "*submit*" yang akan mengarahkan pengguna pada antarmuka simulasi penempatan kamera CCTV.



Input	
Room width	<input type="text" value="800"/> cm
Room length	<input type="text" value="600"/> cm
Camera range	<input type="text" value="600"/> cm
Camera wide angle	<input type="text" value="90"/> °
Max cell size	<input type="text" value="100"/> cm
<input type="button" value="Submit"/>	

Gambar 4.1: Antarmuka penerima masukan

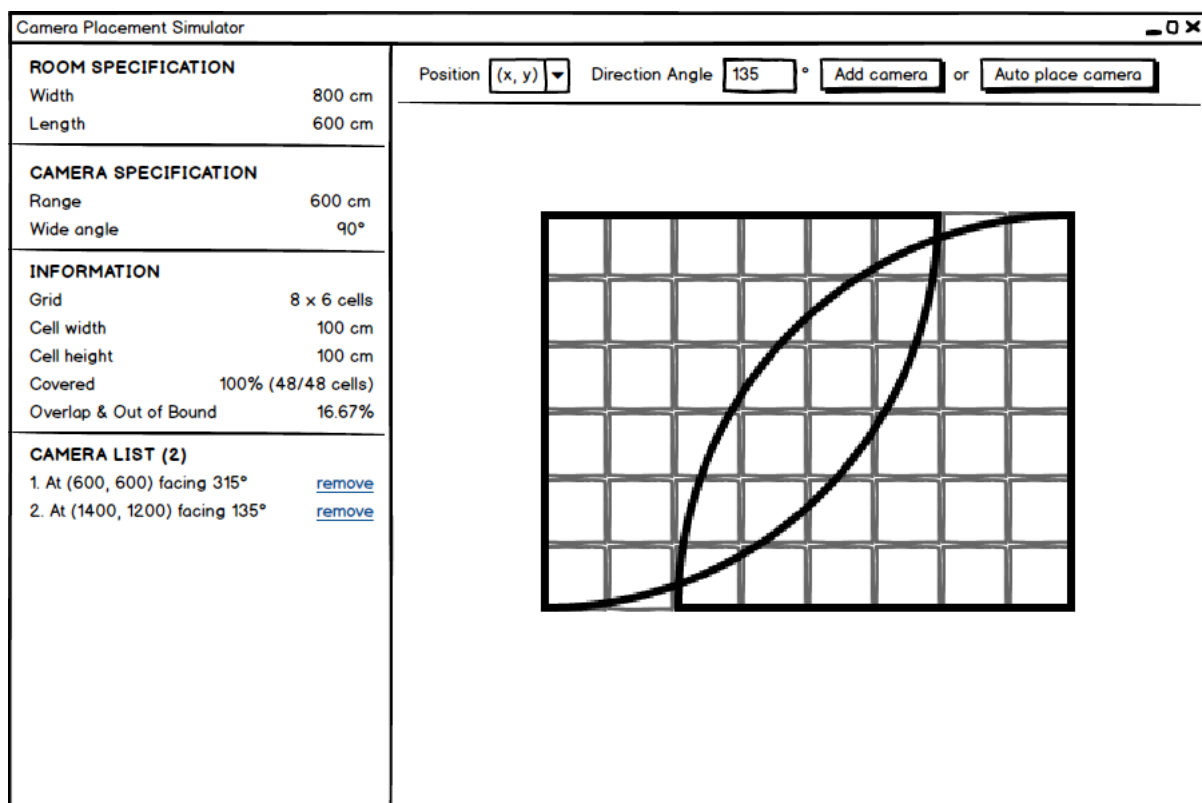
2. Antarmuka: **Simulasi Penempatan Kamera CCTV**

Antarmuka ini berfungsi untuk melakukan kegiatan simulasi penempatan kamera CCTV. Antarmuka ini dapat dilihat pada gambar 4.2. Di dalam antarmuka ini terdapat 3 bagian, yaitu:

- Panel informasi yang berada di bagian kiri antarmuka.
Panel ini berfungsi untuk memberikan informasi-informasi yang terdiri dari ukuran ruangan, spesifikasi kamera CCTV, informasi simulasi, dan daftar penempatan kamera CCTV. Pada bagian informasi simulasi terdapat informasi persentase ketercakupan dan persentase tingkat *overlap* dan *out of bound*. Pada bagian daftar penempatan kamera

CCTV, terdapat penempatan-penempatan yang sedang diterapkan dalam simulasi. Pada setiap penempatan terdapat tombol "*remove*" yang apabila ditekan akan membuang penempatan tersebut.

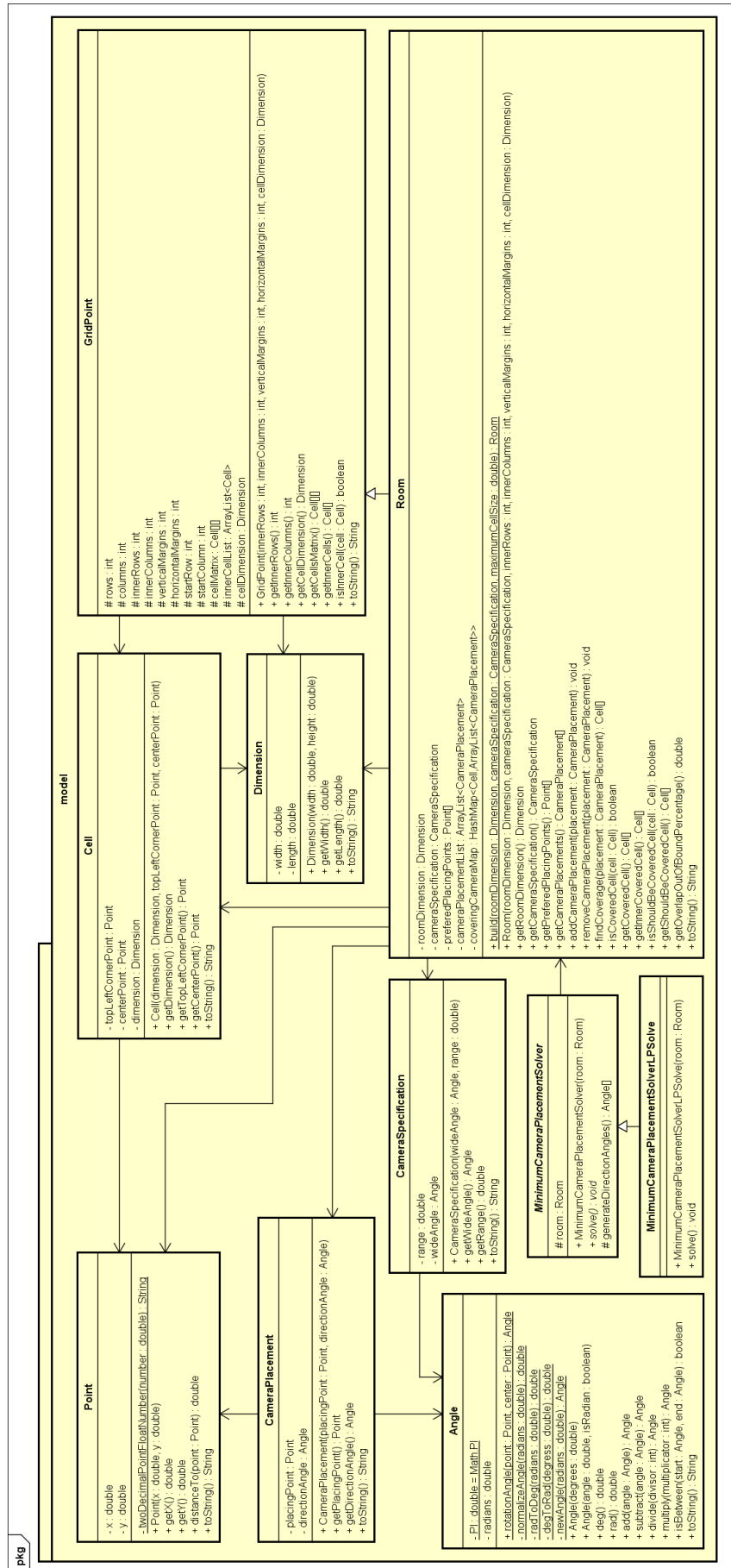
- Panel visualisasi penempatan kamera CCTV yang berada di bagian kanan antarmuka. Panel ini berfungsi untuk menampilkan visualisasi penempatan kamera CCTV sesuai dengan penempatan-penempatan yang sedang diterapkan pada simulasi.
- Panel penambah kamera CCTV yang berada di atas panel visualisasi. Panel ini berfungsi untuk melakukan penempatan kamera CCTV baru. Pengguna dapat melakukan 2 jenis penambahan penempatan, yaitu penambahan sesuai dengan keinginan pengguna dan penambahan secara otomatis. Apabila pengguna ingin melakukan penambahan sesuai dengan keinginan, maka pengguna dapat memilih posisi dan mengisi sudut arah pandang dan dilanjutkan dengan menekan tombol "*add camera*". Apabila pengguna ingin melakukan penambahan secara otomatis, maka pengguna dapat menekan tombol "*auto place camera*". Dengan melakukan penambahan secara otomatis, sistem akan mencari penempatan-penempatan tersedikit yang dapat mencakup seluruh cell yang belum tercakup. Selama proses penambahan otomatis berjalan, tombol "*add camera*" dan tombol "*auto place camera*" akan dinon-aktifkan dan diaktifkan kembali apabila proses telah selesai.



Gambar 4.2: Antarmuka penempatan kamera CCTV

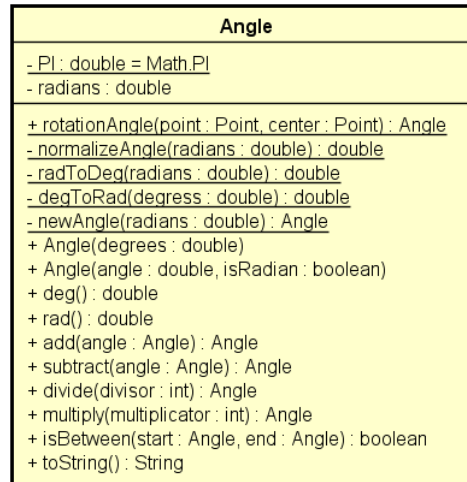
4.2 Perancangan Kelas

Pada bagian ini akan dijelaskan kelas-kelas yang digunakan dalam membangun perangkat lunak. Diagram kelas dapat dilihat pada gambar 4.3.



Gambar 4.3: Diagram kelas lengkap

4.2.1 Kelas *Angle*



Gambar 4.4: Diagram kelas *Angle*

Kelas ini merepresentasikan sudut dan menangani fungsi-fungsi yang berhubungan dengan sudut. Diagram kelas *Angle* dapat dilihat pada gambar 4.4. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *Angle*:

- ***PI* → double**

Atribut ini merupakan atribut statis bernilai π yang dapat digunakan oleh setiap objek dari kelas *Angle*.

- ***radians* → double**

Atribut ini berguna untuk menampung sudut dalam bentuk radian.

Berikut ini merupakan fungsi-fungsi yang terdapat pada kelas *Angle*:

- ***rotationAngle(point → Point, center → Point) → Angle***

Fungsi ini merupakan fungsi statis yang berguna untuk mendapatkan sudut rotasi dari titik *point* terhadap titik *center*.

- ***normalizeAngle(radians → double) → double***

Fungsi ini merupakan fungsi statis yang berguna untuk melakukan normalisasi sudut *radians* sehingga berada dalam rentang $0 \leq radians < 2\pi$.

- ***radToDeg(radians → double) → double***

Fungsi ini merupakan fungsi statis yang berguna untuk mengubah sudut dalam bentuk radian menjadi sudut dalam bentuk derajat.

- ***degToRad(degrees → double) → double***

Fungsi ini merupakan fungsi statis yang berguna untuk mengubah sudut dalam bentuk derajat menjadi sudut dalam bentuk radian.

- ***newAngle(radians → double) → Angle***

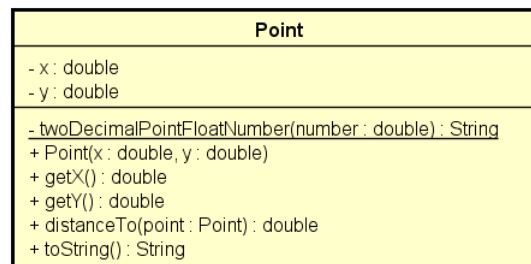
Fungsi ini merupakan fungsi statis yang berguna untuk membuat objek *Angle* baru.

- ***deg()* → double**

Fungsi ini berguna untuk mendapatkan sudut dalam bentuk derajat.

- ***rad()* → *double***
Fungsi ini berguna untuk mendapatkan sudut dalam bentuk radian.
- ***add(angle → Angle) → Angle***
Fungsi ini berguna untuk menghasilkan objek sudut baru yang merupakan hasil penjumlahan antara sudut objek ini dengan sudut objek *angle*.
- ***subtract(angle → Angle) → Angle***
Fungsi ini berguna untuk menghasilkan objek sudut baru yang merupakan hasil pengurangan antara sudut objek ini dengan sudut objek *angle*.
- ***divide(divisor → int) → Angle***
Fungsi ini berguna untuk menghasilkan objek sudut baru yang merupakan hasil pembagian antara sudut objek ini dengan nilai *divisor*.
- ***multiply(multiplier → int) → Angle***
Fungsi ini berguna untuk menghasilkan objek sudut baru yang merupakan hasil pengalihan antara sudut objek ini dengan nilai *multiplier*.
- ***isBetween(start → Angle, end → Angle) → boolean***
Fungsi ini berguna untuk mengetahui apakah sudut objek ini berada di antara sudut objek *start* dan sudut objek *end*.

4.2.2 Kelas *Point*



Gambar 4.5: Diagram kelas *Point*

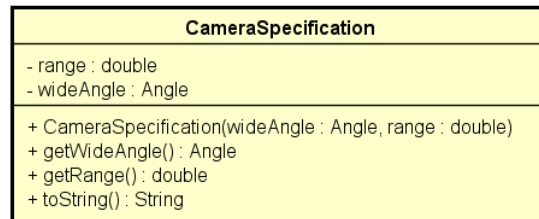
Kelas ini merepresentasikan titik koordinat 2D. Diagram kelas *Point* dapat dilihat pada gambar 4.5. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *Point*:

- ***x* → *double***
Atribut ini berguna untuk menampung nilai titik pada sumbu x.
- ***y* → *double***
Atribut ini berguna untuk menampung nilai titik pada sumbu y.

Berikut ini merupakan fungsi-fungsi yang terdapat pada kelas *Point*:

- ***twoDecimalPointFloatNumber(number → double) → String***
Fungsi ini merupakan fungsi statis yang berguna untuk mengubah bilangan *number* ke dalam bentuk *String* dengan maksimal bilangan di belakang koma berjumlah 2 buah.
- ***distanceTo(point → Point) → double***
Fungsi ini berguna untuk mendapatkan jarak antara titik objek ini dengan titik objek *point*.

4.2.3 Kelas *CameraSpecification*

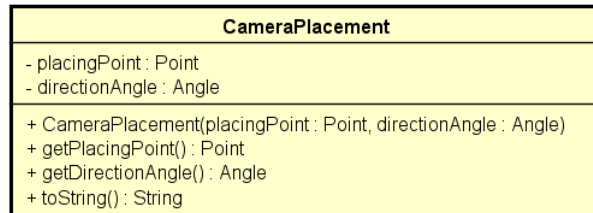


Gambar 4.6: Diagram kelas *CameraSpecification*

Kelas ini merepresentasikan spesifikasi kamera CCTV yang terdiri dari jarak pandang efektif dan besar sudut pandang. Diagram kelas *CameraSpecification* dapat dilihat pada gambar 4.6. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *CameraSpecification*:

- *range* → *double*
Atribut ini berguna untuk menampung jarak pandang efektif kamera CCTV.
- *wideAngle* → *Angle*
Atribut ini berguna untuk menampung besar sudut pandang kamera CCTV.

4.2.4 Kelas *CameraPlacement*

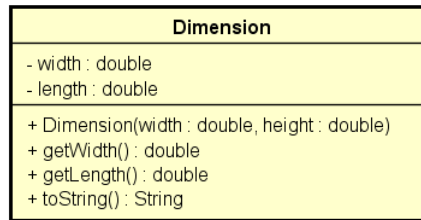


Gambar 4.7: Diagram kelas *CameraPlacement*

Kelas ini merepresentasikan penempatan kamera CCTV yang terdiri dari posisi dan sudut arah pandang. Diagram kelas *CameraPlacement* dapat dilihat pada gambar 4.7. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *CameraPlacement*:

- *placingPoint* → *Point*
Atribut ini berguna untuk menampung posisi penempatan kamera CCTV.
- *directionAngle* → *Angle*
Atribut ini berguna untuk menampung sudut arah pandang kamera CCTV.

4.2.5 Kelas *Dimension*

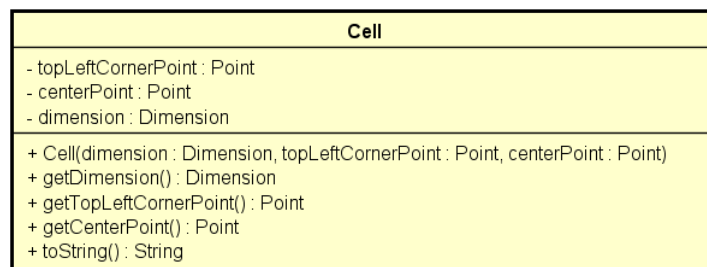


Gambar 4.8: Diagram kelas *Dimension*

Kelas ini merepresentasikan dimensi yang terdiri dari panjang dan lebar. Diagram kelas *Dimension* dapat dilihat pada gambar 4.8. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *Dimension*:

- ***width* → *double***
Atribut ini berguna untuk menampung ukuran lebar.
- ***length* → *double***
Atribut ini berguna untuk menampung ukuran panjang.

4.2.6 Kelas *Cell*

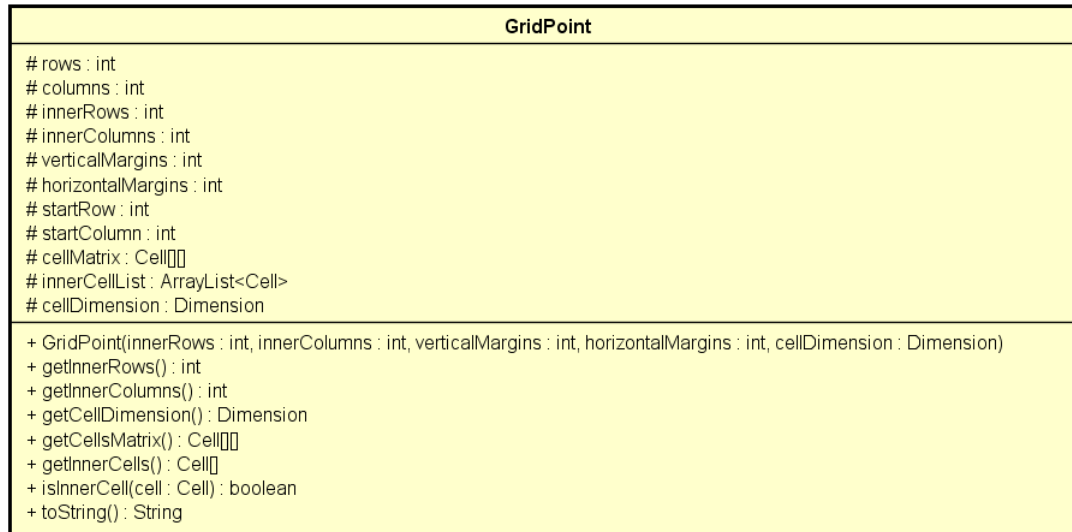


Gambar 4.9: Diagram kelas *Cell*

Kelas ini merepresentasikan cell. Diagram kelas *Cell* dapat dilihat pada gambar 4.9. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *Cell*:

- ***topLeftCornerPoint* → *Point***
Atribut ini berguna untuk menampung titik ujung kiri atas cell.
- ***centerPoint* → *Point***
Atribut ini berguna untuk menampung titik tengah cell.
- ***dimension* → *Dimension***
Atribut ini berguna untuk menampung dimensi cell.

4.2.7 Kelas *GridPoint*

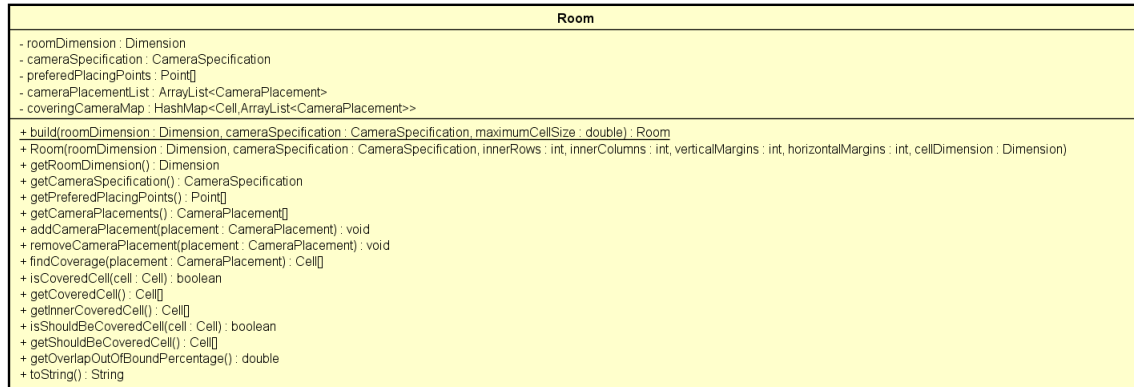


Gambar 4.10: Diagram kelas *GridPoint*

Kelas ini merepresentasikan grid point yang terdiri dari matriks cell. Diagram kelas *GridPoint* dapat dilihat pada gambar 4.10. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *GridPoint*:

- ***rows* → *int***
Atribut ini berguna untuk menampung jumlah baris grid point secara keseluruhan.
- ***columns* → *int***
Atribut ini berguna untuk menampung jumlah kolom grid point secara keseluruhan.
- ***innerRows* → *int***
Atribut ini berguna untuk menampung jumlah baris grid point bagian dalam.
- ***innerColumns* → *int***
Atribut ini berguna untuk menampung jumlah kolom grid point bagian dalam.
- ***verticalMargins* → *int***
Atribut ini berguna untuk menampung jumlah margin vertikal grid point.
- ***horizontalMargins* → *int***
Atribut ini berguna untuk menampung jumlah margin horizontal grid point.
- ***startRow* → *int***
Atribut ini berguna untuk menampung indeks baris pertama dalam grid point bagian dalam.
- ***startColumn* → *int***
Atribut ini berguna untuk menampung indeks kolom pertama dalam grid point bagian dalam.
- ***cellMatrix* → *Cell*[][]**
Atribut ini berguna untuk menampung matriks cell 2 dimensi.
- ***innerCellList* → *ArrayList*<*Cell*>**
Atribut ini berguna untuk menampung cell-cell yang berada pada grid point bagian dalam.
- ***cellDimension* → *Dimension***
Atribut ini berguna untuk menampung dimensi cell.

4.2.8 Kelas *Room*



Gambar 4.11: Diagram kelas *Room*

Kelas ini merepresentasikan ruangan yang dapat diisi oleh kamera-kamera CCTV. Kelas ini merupakan turunan dari kelas *GridPoint*. Diagram kelas *Room* dapat dilihat pada gambar 4.11. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *Room*:

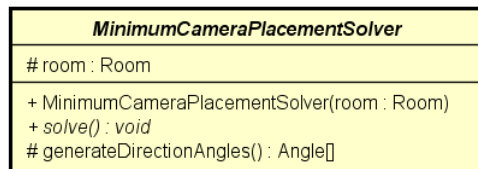
- ***roomDimension* → *Dimension***
Atribut ini berguna untuk menampung dimensi ruangan.
- ***cameraSpecification* → *CameraSpecification***
Atribut ini berguna untuk menampung spesifikasi kamera CCTV yang akan ditempatkan dalam ruangan.
- ***preferredPlacingPoints* → *Point*[]**
Atribut ini berguna untuk menampung posisi-posisi yang dapat ditempati oleh kamera CCTV.
- ***cameraPlacementList* → *ArrayList*<*CameraPlacement*>**
Atribut ini berguna untuk menampung daftar penempatan kamera CCTV.
- ***coveringCameraMap* → *HashMap*<*Cell*, *ArrayList*<*CameraPlacement*>>**
Atribut ini berguna untuk menampung pemetaan cell dengan penempatan-penempatan kamera CCTV yang dapat mencakup cell tersebut.

Berikut ini merupakan fungsi-fungsi yang terdapat pada kelas *Room*:

- ***build*(*roomDimension* → *Dimension*, *cameraSpecification* → *CameraSpecification*, *maximumCellSize* → *double*) → *Room***
Fungsi ini merupakan fungsi statis yang berguna untuk membuat objek *Room* yang dimana jumlah baris, jumlah kolom, jumlah margin vertikal, jumlah margin horizontal, dan ukuran cell akan ditentukan berdasarkan ukuran ruangan *roomDimension*, spesifikasiKamera *cameraSpecification*, dan ukuran terbesar cell *maximumCellSize*.
- ***addCameraPlacement*(*placement* → *CameraPlacement*) → *void***
Fungsi ini berguna untuk menambahkan penempatan kamera CCTV ke dalam daftar penempatan kamera CCTV dan memperbaharui pemetaan cell pada atribut *coveringCameraMap*.
- ***removeCameraPlacement*(*placement* → *CameraPlacement*) → *void***
Fungsi ini berguna untuk membuang penempatan kamera CCTV dari daftar penempatan kamera CCTV dan memperbaharui pemetaan cell pada atribut *coveringCameraMap*.

- ***findCoverage(placement → CameraPlacement) → Cell[]***
Fungsi ini berguna untuk mendapatkan cell-cell yang tercakup oleh penempatan *placement*.
- ***isCoveredCell(cell → Cell) → boolean***
Fungsi ini berguna untuk mengetahui apakah cell *cell* telah tercakup oleh setidaknya 1 penempatan kamera CCTV.
- ***getCoveredCell() → Cell[]***
Fungsi ini berguna untuk mendapatkan cell-cell yang telah tercakup oleh setidaknya 1 penempatan kamera CCTV.
- ***getInnerCoveredCell() → Cell[]***
Fungsi ini berguna untuk mendapatkan cell-cell yang berada pada grid point bagian dalam dan telah tercakup oleh setidaknya 1 penempatan kamera CCTV.
- ***isShouldBeCoveredCell(cell → Cell) → boolean***
Fungsi ini berguna untuk mengetahui apakah cell *cell* berada pada grid point bagian dalam dan belum tercakup oleh setidaknya 1 penempatan kamera CCTV.
- ***getShouldBeCoveredCell() → Cell[]***
Fungsi ini berguna untuk mendapatkan cell-cell yang berada pada grid point bagian dalam dan belum tercakup oleh setidaknya 1 penempatan kamera CCTV.
- ***getOverlapAndOutOfBoundPercentage() → double***
Fungsi ini berguna untuk mendapatkan persentase *overlap* dan *out of bound*.

4.2.9 Kelas *MinimumCameraPlacementSolver*



Gambar 4.12: Diagram kelas *MinimumCameraPlacementSolver*

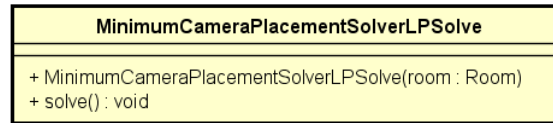
Kelas ini merepresentasikan pemecah masalah penempatan kamera CCTV dalam ruangan agar berjumlah minimum. Kelas ini merupakan kelas abstrak. Diagram kelas *MinimumCameraPlacementSolver* dapat dilihat pada gambar 4.12. Berikut ini merupakan atribut-atribut yang terdapat pada kelas *MinimumCameraPlacementSolver*:

- ***room → Room***
Atribut ini berguna untuk menampung ruangan yang akan diselesaikan masalahnya.

Berikut ini merupakan fungsi-fungsi yang terdapat pada kelas *Room*:

- ***solve() → void***
Fungsi ini merupakan fungsi abstrak yang bertujuan untuk menyelesaikan masalah penempatan kamera CCTV dalam ruangan agar berjumlah minimum.
- ***generateDirectionAngles() → Angle[]***
Fungsi ini berguna untuk menghasilkan sudut-sudut arah pandang yang akan digunakan dalam penyelesaian masalah.

4.2.10 Kelas *MinimumCameraPlacementSolverLPSolve*



Gambar 4.13: Diagram kelas *MinimumCameraPlacementSolverLPSolve*

Kelas ini merepresentasikan pemecah masalah penempatan kamera CCTV dalam ruangan agar berjumlah minimum dengan menggunakan kakas *lpsolve*. Kelas ini merupakan turunan dari kelas *MinimumCameraPlacementSolver*. Diagram kelas *MinimumCameraPlacementSolverLPSolve* dapat dilihat pada gambar 4.13. Berikut ini merupakan fungsi-fungsi yang terdapat pada kelas *MinimumCameraPlacementSolverLPSolve*:

- ***solve()* → void**

Fungsi ini berguna untuk menyelesaikan masalah penempatan kamera CCTV dalam ruangan agar berjumlah minimum dengan menggunakan kakas *lpsolve*.

DAFTAR REFERENSI

- [1] de Berg, M., Cheong, O., van Kreveld, M. J., dan Overmars, M. (2008) *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, Berlin.
- [2] van Kreveld, M. J. (2004) Geographic information systems. Bagian dari Goodman, J. E. dan O'Rourke, J. (ed.), *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC, Boca Raton.
- [3] Buchin, K., Buchin, M., van Kreveld, M. J., Löffler, M., Silveira, R. I., Wenk, C., dan Wiratma, L. (2013) Median trajectories. *Algorithmica*, **66**, 595–614.
- [4] van Kreveld, M. J. dan Wiratma, L. (2011) Median trajectories using well-visited regions and shortest paths. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, USA, 1-4 November, pp. 241–250. ACM, New York.
- [5] Lionov (2002) Animasi algoritma sweepline untuk membangun diagram voronoi. Skripsi. Universitas Katolik Parahyangan, Indonesia.
- [6] Wiratma, L. (2010) Following the majority: a new algorithm for computing a median trajectory. Thesis. Utrecht University, The Netherlands.
- [7] Wiratma, L. (2022) Coming Not Too Soon, Later, Delay, Someday, Hopefully. Disertasi. Utrecht University, The Netherlands.
- [8] van kreveld, M., van Lankveld, T., dan Veltkamp, R. (2013) Watertight scenes from urban lidar and planar surfaces. Technical Report UU-CS-2013-007. Utrecht University, The Netherlands.
- [9] Rekhter, Y. dan Li, T. (1994) A border gateway protocol 4 (bgp-4). RFC 1654. RFC Editor, <http://www.rfc-editor.org>.
- [10] ITU-T Z.500 (1997) *Framework on formal methods in conformance testing*. International Telecommunications Union. Geneva, Switzerland.
- [11] Version 9.0.0 (2016) *The Unicode Standard*. The Unicode Consortium. Mountain View, USA.
- [12] Version 7.0 Nougat (2016) *Android API Reference Manual*. Google dan Open Handset Alliance. Mountain View, USA.
- [13] Webb, R., Daruca, O., dan Alfadian, P. (2012) *Method of optimizing a text message communication between a server and a secure element*. Paten no. EP2479956 (A1). European Patent Organisation. Munich, Germany.
- [14] Wiratma, L. (2009) Median trajectory. Report for GMT Experimentation Project at Utrecht University.
- [15] Lionov (2011) Polymorphism pada C++. Catatan kuliah AKS341 Pemrograman Sistem di Universitas Katolik Parahyangan, Bandung. <http://tinyurl.com/lionov>. 30 September 2016.

- [16] Erickson, J. (2003) CG models of computation? <http://www.computational-geometry.org/mailling-lists/compgeom-announce/2003-December/000852.html>. 30 September 2016.
- [17] AGUNG (2012) Menjajal tango 12. Majalah HAI no 02, Januari 2012.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```
1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

Listing A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```


LAMPIRAN B

HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4